A Study of Issues in Multiconnected Distributed Loop Interconnection Networks

A thesis submitted in partial fulfilment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in

ENGINEERING

by

Nimmagadda Chalamaiah

under the supervision of

Dr. R.Badrinath

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY
KHARAGPUR 721 302 INDIA

July 1999

In memory of my parents

Dr. Sethu Madhava Rao
Sarojini

<u>CERTIFICATE</u>

This is to certify that the thesis entitled **A Study of Issues in Multiconnected Distributed Loop Interconnection Networks**, submitted by **Nimmagadda Chalamaiah** for the award of the degree of **Doctor of Philosophy** is a record of original research work carried out by him under my supervision and guidance. The thesis has fulfilled all requirements as per the regulations of the institute and in my opinion has reached the standard needed for submission. Further, the results embodied in this thesis have not been submitted to any other university or institute for the award of any degree or diploma.

Dr. R.Badrinath,
Assistant professor,
Department of Computer Science and Engineering,
Indian Institute of Technology,
Kharagpur, 721 302.
India.

ACKNOWLEDGEMENTS

I am deeply indebted to Dr. R.Badrinath, Department of Computer Science and Engineering, Indian Institute of Technology, Kharagpur who not only is my research supervisor but an ever lasting guiding-spirit all through my work. His special attention towards my work always made me to work on new ideas in a simple and direct way of solving them. Several anonymous reviewers of my research papers that were presented in national and international conferences, and published in journals provided me invaluable feedback concerning organisation, coverage and presentation.

My sincere thanks are due to Prof. A.K.Majumdar, Head, and Prof. Ajit Pal, former Head of this department, for their encouragement and understanding. I also thank Dr. I.Sengupta and Dr. Prabir K.Biswas for their valuable suggestions. I specially thank Prof. Sujay Ghosh and Dr. D. Sarkar for their help as and when I approached them. I put on record the co-operation I received from the other faculty of this department. I will be failing in my duty if I do not mention the laboratory staff and administration of this department for their timely help.

The infrastructural facilities provided in the laboratory and library of the department during pre-registration and research periods are gratefully acknowledged. Also I am indebted to All India Council for Technical Education for awarding me Quality Improvement Programme scholarship. I wish I would never forget the company I had from my fellow research scholars: Frahad Musadeekh, Jatindra Kumar Deka, Biswajit Acharya, Parag Kulkarni, Diganta Goswami, Rajdeep Niyogi, R.Parthasarathi from my department and a memorable time with M.C. Nataraja of civil engineering department.

I wish to put on record the enormous debt of gratitude I owe to my colleagues at JNTU College of Engineering Kakinada for having shared work load in the sabbatical period of my work.

My heart is filled with affectionate gratitude to my wife Annapoorna for the inspiration and moral support she provided throughout the long period of my research work and whose patience was tested to the utmost by my erratic working hours and a long period of separation. Without her loving support and understanding I would never have completed my present work. Particularly I owe to my daughter Padmavathi who really allowed me to do what I want and looked after herself for all that she wanted from a father. I can only say to them "Here we are, at last."

Most importantly the one "who had been there" when I needed him is Dr. I.V.Rao, M.D, my family friend and my special thanks are due to him.

14th July, 1999. **N.Chalamaiah**

Abstract

In this thesis we present results of our investigations pertaining to multiconnected distributed loop interconnection network (MCDL) topology. We focus on the following key issues of MCDL in both the shared medium and direct network applications.

1. Algorithms for computing inter node distance, diameter and average path length.
2. Packet transfer delays in MCDL when used in shared medium applications with multiple token media access protocol.
3. Routing algorithms under node and edge faults when the faulty component location is known globally or in the neighbourhood only.
4. Deadlock-free routing algorithms using virtual channels in direct network multicomputer interconnection applications.

The inter node distance, diameter and average path length in MCDL are important factors in the design of an interconnection network. It is an open problem to obtain a closed form expressions for these parameters in MCDL. Although we also do not have a closed form, we give algorithms to compute these parameters which t run in a time that is independent of diameter and number of node in MCDL. From these algorithms we tabulate optimal loops, their diameters and average path lengths. Some properties of these optimal loops are conjectured from these results. We also discuss and try to explore why these parameters have no closed form expressions.

In the study of multiple token media access communication protocol, we consider three routings: *hop to loop*, *loop to hop* and *no switching*. In these three routings, packet transfer delays are obtained analytically for different values of hop sizes using queueing theory techniques. These results are compared with single token and multiple ring delays. This study shows the effect of hop size of MCDL and intermediate stage queueing in the media access protocol. We have developed a discrete event simulator to verify our analytic results and assumptions there in.

While developing algorithms for the fault-tolerant routing in MCDL, we consider node and edge faults. We consider the cases that the address of the faulty node or edge is either globally available to all the nodes in MCDL or only to the nodes with in the neighbourhood of the fault. We conclude that the shortest path increases is by no more than two in either of the cases of the fault knowledge.

We study MCDL in wormhole routed direct networks applications. In wormhole routing, packets are forced to travel through intermediate nodes which can lead to *dependency* loops and consequently *deadlocks*. Deadlocks in direct networks are avoided using *virtual channels*. We consider *directed ring*, *undirected ring* and *MCDL* topologies, and first show that they have cyclic channel dependencies. Later we introduce virtual channels to remove cyclic channel dependencies. Then the hop to loop and loop to hop routings in MCDL are shown to be deadlock-free.

Chapter 1

INTRODUCTION

The performance of most digital systems today is limited by their *communication* and *interconnection* and not by their logic or memory [DYN97]. As technology improves, on one side the memories and processors are becoming small, fast and inexpensive, but on the other side the frequency of communication between the components is lagging far behind the clock rates of modern processors. Designers strive to make more efficient use of scarce interconnection bandwidth, pin density and wiring density that govern interconnections between systems. These factors combine to make *interconnection* a key factor in the success of future digital systems. Interconnection networks are emerging as nearly universal solution to the system level communication problems for modern digital systems. They are beginning to replace buses as the standard system-level interconnection. They are also replacing dedicated wiring in special purpose systems as designers discover that routing of packets is both faster and more economical than routing wires. Interconnection networks are being used for many different applications: back plane buses, telephone switches, internal networks for asynchronous transfer mode (ATM) switches, processor/memory interconnects for vector supercomputers, cluster of workstations, multicomputers, distributed shared memory multiprocessors, local area networks, metropolitan area networks and wide area computer networks.

The number of applications requiring interconnection networks is continuously growing. They have advanced into all the areas of architectures and environments. More recently, this network technology began to be transferred to local area networks (LAN) also and has been proposed as a replacement for back plane buses, creating the concept of *system area networks.* Thus when this technology has advanced into different environments, there is a need for structuring the concepts, solutions and also a need arises to address common issues and its topological properties. In our work we mainly develop studies of interconnection networks related to topological properties and in the application areas of local area networks and multicomputers. We will also present performance studies/results specifically for these two environments.

1.1 Classification of interconnection networks

Interconnection networks have traditionally been classified according to operating mode (asynchronous or synchronous) and network control (centralised, decentralised or distributed). But we focus on other criteria for classification that are currently more significant. Our classification is primarily based on network topology as in [DYN97]: *shared- medium networks*, *direct networks*, *indirect networks* and *hybrid networks*. In shared-medium networks, the transmission medium is shared by all communicating devices. An alternative approach consists of having point-to-point links directly connecting each communicating device to a subset of other communicating devices requiring all communications between non-neighbouring devices to be through several intermediate devices. These networks are known as *direct networks*. In indirect networks, instead of directly connecting the communicating devices, they are connected through one or more switches. The switches are connected between them using point-to-point links. Hence any communication between the communicating devices pass through one or more switches. Finally, some hybrid approaches of the above three are also possible. In our work we study interconnection networks in the application areas of shared medium and direct network environments.

1.2 Shared medium networks

Referring to the classification of interconnection networks given in Section 1.1, shared medium networks are least expensive as the transmission medium is shared by all the communicating devices. In such networks, only one device is allowed to use the network at a time. The network itself is usually passive. There are two major classes of shared medium networks: *local area networks (LAN)*, mainly used to construct computer networks that span physical distances no longer than a few kilometers, and *back plane buses*, used mainly for internal communication in uniprocessors and multiprocessors. High sped LANs can be used as networking backbone to interconnect computers to provide an integrated parallel and distributed environment. An important issue in shared medium networks is the *arbitration strategy* that determines the mastership of the shared medium to resolve network access conflicts. For performance and implementation reasons, it is impractical to have either a *centralised control* or some *fixed assignment* to determine the mastership to access the medium.

Depending on the arbitration mechanism used, different LANs have been commercially available. Three major classes of LANs based on distributed control are: *contention bus*, *token bus* and *token ring*. In contention bus, all the devices connected to the shared medium (the bus) compete for the exclusive right to access. Due to this a "collision" is likely to occur. When collision is detected the competing devices will quit and retry later. The most well known contention based LAN is Ethernet which adopts carrier-sense multiple access with collision detection (CSMA/CD). When processing speed becomes faster, the number of devices that can be connected to Ethernet is limited to avoid network bottleneck. Moreover one draw back of CSMA/CD is the nondeterministic nature of the access right. Hence such buses are not suitable for real-time applications. To remove the nondeterministic behaviour, in a token bus a token is passed among the network devices. The owner of the token has the right to access the bus. Upon completion of transmission, the token is passed to the next device. The holding time of the token affects the throughput of token bus networks. The idea of a token ring is a natural extension of token bus. The IBM token ring supports a bandwidth of 4 to 16 Mbps on coaxial cable and Fiber Distributed Data Interface FDDI is a token ring which provides a bandwidth of 100 Mbps.

1.3 Direct networks

Scalability is an important factor in multiprocessor systems. Bus based systems are not scalable as the bus becomes the bottleneck when more processors are added. Direct network topologies are promising candidates for interconnection of computers in multiprocessor and multicomputer architectures. In a direct network architecture each node has direct point-to-point connections to some number of other nodes called neighbouring nodes. Most commonly used topologies for direct networks are *k_ary_n cube, mesh, tori and hypercube.* Each node is a small programmable computer with its own processor, local memory and other supporting devices. A common component in a node is a *router* to handle communications among the nodes. Although the functions of a router can be performed by the local processor, a dedicated router allows overlapped computation and communication within the node. For this reason direct networks are also called *router based* networks. Nodes in the network communicate by passing messages whose size may vary depending on the application. Messages are often divided into fixed size packets. A packet carries the destination address and sequence number along with the data. The routing algorithms determine the path taken by a packet to reach its destination. When a packet header reaches an intermediate node, a *switching mechanism* called *wormhole switching* determines how and when the router switch is set *i.e.*, the input channel is connected to the output channel selected by the routing algorithm. Obviously some buffer space is required to store the packet until the next channel is reserved. In order to make the buffer requirements simple, a message is broken into flits and is pipelined in the network at the flit level. The main advantages of wormhole switching are low memory requirements in routers, pipelined data movement and low communication latency. But the main disadvantage of wormhole switching is *deadlocks.* As each packet whose header has not reached its destination requests some buffers while keeping the buffers currently storing the packet. If packets can not advance because the buffers requested by them are full, then deadlocks occur. Therefore when a deadlock occurs among a set of packets, every packet in that set is requesting resources held by other packets while holding resources requested by other packets of the set.

In our work we study and propose deadlock-free routing algorithms in direct networks.

1.4 Performance metrics of interconnection networks

Very large and complex interconnection networks have been used in the design of distributed computer systems, inter-PE (processor element) communication networks in parallel computing environment and local area networks, to perform necessary data routing and manipulation functions. Distributed systems are an attractive alternative to centralised systems from the reliability view point. In a distributed system the processing, control and databases are distributed over computers that are interconnected. An important component of a distributed system is the topology for interconnection of computers. Also there are well defined relations between system topology and the message delay, the routing algorithms and, the fault tolerance and recovery. Specifically the message delay may be directly proportional to the inter node distances; the complexity of a routing algorithm may be determined by the topology; and the fault tolerance/recovery capabilities may be directly related to the node connectivity.

In our work we address the performance metrics related to topology such as diameter and average path length. We also address the metrics concerning the delay, fault tolerance and deadlock-free routing.

1.5 Motivation

We motivated our work in the study of an interconnection network topology that is suitable for both the shared medium and direct network applications. The network we considered for our study is *multiconnected distributed loop (MCDL)* network. In a *multiconnected distributed loop network* of n nodes every node i is connected to $(i + h)$, $(i - h)$, $(i + 1)$ and $(i - 1)$ mod n, where h is called the *hop length* or *skip distance*. These connections from node i are also termed as $[+h]$, $[-h]$, $[+1]$ and $[-1]$ links. MCDL interconnection networks are extensions of ring topologies. Some commercial multiprocessor systems such as ILLIAC-IV, KSR-1 and CDC-Cyberplus are designed keeping the ring topology as interconnection network in some form [HB89, Hwa93].

We present our studies in MCDL in the aspects of: topological properties such as diameter, average distance, performance characteristics such as packet transfer delays when used in shared medium applications, fault tolerance when node or edge in MCDL fails and deadlock-free wormhole routing algorithms with virtual channels in direct network applications. In the following we present the scope of our study on the above four aspects of MCDL interconnection network.

1.5.1 Graph properties of MCDL

In [MS95], computation of shortest path between a pair of nodes was reported in $O(\delta)$ time, where δ is the diameter of MCDL. In [DH90, Tzv91, BT91], the optimal forms of n for which the diameter is minimum over all possible hop sizes were reported for only specific values of n. We find that there exist scope to have a faster algorithm for the shortest path using *number theoretic functions* [Ste65], find the diameter and average distance between the nodes and the corresponding optimal values for all values of n. In our work, we develop an algorithm for finding the shortest path that is faster than the one in [MS95] and apply it for finding diameter, average distance between nodes and the corresponding optimal values of n.

1.5.2 Token based communications in MCDL

We find that $MCDL$ can also be a suitable topology for *multiple token* operation in shared medium applications in view of the ring topology being embedded in it. In the $MCDL$ topology, we consider a directed version with M nodes and hop size h. With such a configuration, there exist g number of *hop rings* where $g = GCD(M, h)$ in the topology. A hop ring is one that is formed by the connections node i to node $(i + h)$ mod M. Also there exist a *loop ring* with connections node i to node $(i + 1)$ mod M. So in all the $(g + 1)$ rings we assume that a token exists in each of them and medium access rights are granted through these token(s). The transmission of the packet along with the token follows the shortest path. The shortest path is either a combination of both hop and loop ring traversals or only one of them. In other words, some packets need both the hop and loop ring tokens, while some other packets need only one of the hop/loop ring tokens. In the former case, the routing needs *switching* of the packet from hop to loop or loop to hop depending on the routing implementation. Thus we have two routing schemes for the packets: *hop to loop routing* and *loop to hop routing*. We also can have a case of routing under *no switching*, where a packet does not switch but takes a path of either only hop or only loop traversals depending on the destination node address.

We analyse $MCDL$ system with multiple tokens under these three routing schemes, for various values of hop sizes and packet lengths using *queueing* models [Kle75] for the packet delay. We developed a discrete event simulator in our work to verify the analytic results.

1.5.3 Routing under faults in MCDL

From the shortest path result, we found that there exist multiple such paths between a source destination pair. In view of this, an insight into the fault tolerant studies throw good results in direct network applications. In our work we primarily consider two cases of network state under faults: *global knowledge of faults* and *local knowledge of faults*, where the occurrence of fault (node or edge fault) and its corresponding physical address in the network is available at all other nodes in the system or only at the adjacent nodes respectively. Infact each of the cases has its own overhead in reporting faults throughout the network. However we do not study these overheads. But what we report in our work is how efficiently a packet can be routed in a faulty MCDL under the two cases of network states mentioned above. We also present cases that arise which increases the shortest path length under some special cases of faults.

1.5.4 Deadlock-free wormhole routing in MCDL

In order to regard MCDL for deadlock-free wormhole routing in a direct network application, we apply the theory developed in [DS87] for deadlock detection and a flow control called *virtual channel flow control* developed in [Dal92]. The use of virtual channels in a given direct interconnection network transforms it into a *virtual interconnection* network. It is proved that use of virtual channels avoid deadlocks [DS87]. Our work presents a subtle difference in the treatment given here and earlier research [BC96, DA93] for deadlock-free routing. In [BC96, DA93], a routing function was defined for a virtual interconnection network and then a channel dependency graph was derived from this routing by using it as an implicit definition of all possible connections [DS87]. In contrast, we take the channel dependency graph as the definition of what connections are possible and any routing function developed for the network are restricted to use only those connections. This approach seems to highlight a separation between structure of virtual interconnection networks and the routing algorithms that indicate how messages will travel [LH91]. Thus, instead of showing each new routing function deadlock-free, the virtual interconnection network is shown to be deadlock-free for a given channel dependency graph and the new routing function are shown to obey the restrictions of the channel dependency graph.

1.6 Organisation of the thesis

In **Chapter 2** of the thesis we introduce terminology and definitions, and briefly review previous work so that the background for the present work can be understood.

In **Chapter 3**, we give two theorems for finding number of $[+h]$ links between a node pair. Using these results we give an algorithm for shortest path between a pair of nodes in $O(\frac{h}{g} + \log h)$. We note that this running time is independent of number of nodes, edges or diameter of the network.

In **Chapter 4**, we use the algorithms of shortest path given in Chapter 3 to tabulate *diameter, average distances* and their optimal values for different values of number of nodes and hop sizes. We show their non-linearity with number of nodes. We also show from the tables that *average path optimal* loops are also *diameter optimal* but the converse is not true. We compare MCDL with other topologies such as *k-ary n-cube, mesh* and *tori* in terms of diameter, degree and number of nodes.

The **Chapters 5, 6 and 7** are devoted to *token based communications* in MCDL.

In **Chapter 5**, we explain the *multitoken MAC protocol.* This protocol and the shortest path results of Chapter 3 together are applied to define three routings: *hop to loop (HTL), loop to hop (LTH) and no-switching (NOS).* We give the internal structure of a node with buffers and the buffer queues with $M/G/1$ model for the three routings. We derive a result here that the *switch over* packets have Poisson distribution.

In **Chapter 6**, we use $M/G/1$ traffic waiting time results to find delays of packets when $g = 1$, $g = h$ and $1 < g < h$ (where $g = GCD(n, h)$) under HTL, LTH and NOS routings, exponentially distributed packet lengths and for various values of fixed packet lengths. Comparisons of delays are drawn between single ring [SS85], multiple rings [BGY89] and MCDL. Discussions are presented at the end of these comparisons.

In **Chapter 7**, simulation results for all the three routings and packet lengths are obtained. Delay comparisons are drawn between the analysis and simulations.

In **Chapter 8**, fault tolerant routing in MCDL under node and edge faults are explained. We consider both *global* knowledge of faults and *neighbourhood* knowledge on faults.

In **Chapter 9**, we investigate MCDL as a *direct network* topology in multicomputer interconnection networks. After describing *wormhole* routing in multicomputers, we describe the possibility of *deadlocks* in wormhole routing in MCDL. We explain the use of *virtual channels* for avoiding deadlocks in MCDL. Then we present *deadlock-free* wormhole routing in MCDL.

Finally in **Chapter 10**, we summarise the current work and sketch future directions for investigation.

Chapter 2

SURVEY

Ring networks have large diameter and path lengths. Multiconnected loop networks are formed by rings with additional hop connections. They offer not only smaller diameter and path lengths, but also have better fault tolerance. Multiconnected loops have been studied for finding shortest path between nodes, their diameter, average path length and fault tolerance. These parameters are of importance in distributed computing environment having network of processors and in multiprocessor interconnection networks. Ring networks are also popular in *token based* local area networks. Multiconnected loops can operate on multiple token based access protocols and offer better bandwidth sharing. Multiconnected loops have also been studied as direct network topologies for wormhole routing. We study deadlock-free wormhole routing algorithms in direct networks. In the present Chapter we survey related work in these areas.

2.1 Loop topologies

A ring network of n processors, labelled $0, 1, 2, \ldots n-1$, is formed with processor i being connected to processor $(i+1) \bmod n$ and is called a *directed ring* as the edges are directed. Directed rings are vulnerable to faults and have larger communication delays. Failure of a node or a link disrupts the communication among all the remaining nodes. The diameter of directed rings is $n-1$, which results in larger communication delays. One simple way to improve their diameter is by additionally connecting node i to node $(i-1) \bmod n$ also. Then the ring becomes an *undirected* or *doubly connected loop* providing alternate path in case of link failure. But node failures still disrupt the communication. Moreover the diameter can be improved to only $\frac{n}{2}$. Systematic addition of links to this doubly connected loop can improve its performance. One such way is to connect every node i in the doubly connected loop to $(i+h)$ and $(i-h)$ mod n also: here h is called the *hop length* or *skip distance*. Such networks called *multiconnected distributed loop (MCDL) networks* are extensions of ring topologies. Loops having 20 nodes and hop sizes of 3, 5 and 6 are shown in Figures-2.1, 2.2 and 2.3 respectively. These are widely used in design and implementation of distributed systems and local area networks [Par95, HK94, RS86, RGA85].

2.1.1 Ring networks in distributed systems

Distributed systems are an attractive alternative to centralised systems from the reliability view point [Liu78]. In a distributed system the processing, control and databases are distributed over computers that are interconnected. An important component of distributed system is the topology for interconnection of computers [Wil72]. Also there are well defined relations [EH92] between system topology and the message delay, the routing algorithms, and the fault tolerance and recovery. Specifically the message delay is directly proportional to the inter node distances, the complexity of a routing algorithm is determined by the topology and the fault tolerance and recovery capabilities are directly related to the node connectivity. Several topologies have been proposed in [Fen81, Ren80] to this end.

Some commercial multiprocessor systems have been built around ring topology. In fact the ILLIAC-IV, KSR-1 and CDC-Cyberplus are designed with the ring topology as interconnection network in some form [HB89, Hwa93]. ILLIAC-IV uses ring interconnection of 64 processors under one controller for massively parallel processing. CDC-Cyberplus used a hierarchical 2-level slotted token ring (to interconnect up to 34 nodes) for building a massively parallel processor. In 1992 Kendall Square Research used hierarchical 2-level ring for their KSR-1 model. Their architecture is a Cache Only Memory Architecture (COMA) called ALLCACHE by the KSR engineers. The ring in KSR-1 is a pipelined or packet switched 2-level architecture. It supports 8 to 1088 processor memory pair interconnection and every known form of parallelism was supported via KSR's Mach-based operating system.

2.1.2 Ring networks in LAN systems

Ring network architectures present attractive solution to local area networking because of the simplicity of the network interfaces. They permit efficient operation at higher data rates and over larger distances than broadcast buses [Bux81]. An important issue in ring networks is the control mechanism used for message transmission. This mechanism can be centralised or distributed and a distributed mechanism seems to be more advantageous in terms of reliability and performance. A popular centralised mechanism is *slotted access* [Pie72]. A commonly used distributed control mechanism is *control token access* [FN69]. The *delay insertion mechanism* [Liu78] combines both these features. In slotted access, the communication channel is divided into fixed time slots. A node places data in these slots. This mechanism adds overhead of dividing the packets at source to fit into the slots and again assembling at the destination. In

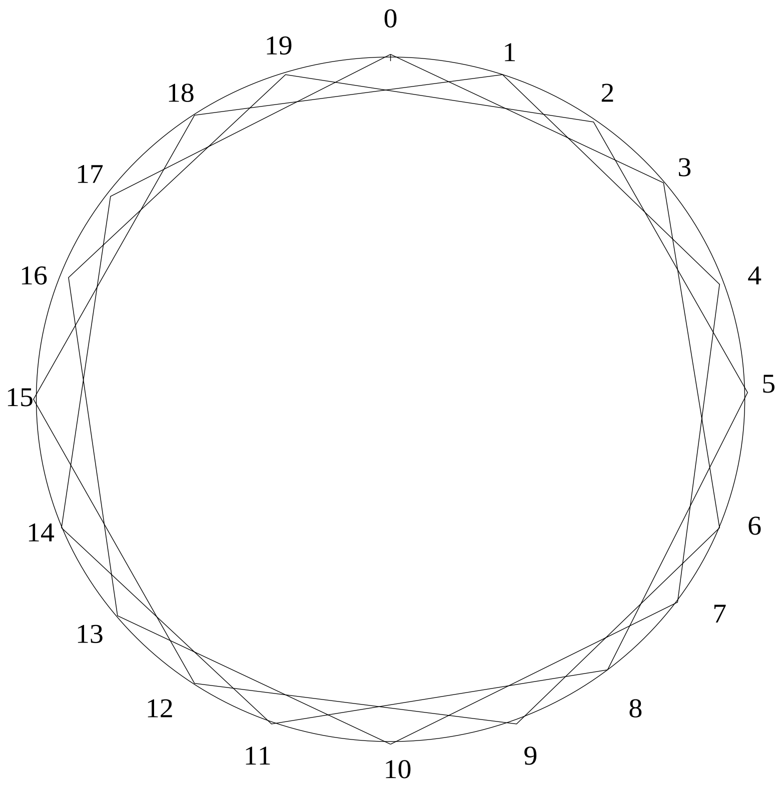


Figure 2.1: Distributed loop with $n = 20$ and $h = 3$

control token access, a token circulates around the ring and a node in possession of it transmits a message. This mechanism adds token latency to the queueing delay. In delay insertion register, a node has a variable length shift register to delay the incoming messages during the transmission of locally generated messages from its insertion buffer. This mechanism adds the overhead of checking address of each received message and also careful conflict resolution is to be made when messages from neighbourhood node and local host arrive the node interface simultaneously. Bux has analysed and compared these three in [Bux84] and finds that the control token access method, which has a distributed loop control, has better delay performance than the other two at high loads.

Some of the important issues in the design of ring networks in local area networks are reliability and delay. One approach to improve the reliability and decrease the delay is by link redundancy [RS86]. We explore the suitability *multiconnected distributed loop network* in these contexts. We propose multiconnected distributed loop networks in local area networks. Since the MCDL has ring structures in it, we propose multi token media access for the study of delay performance [CB98a].

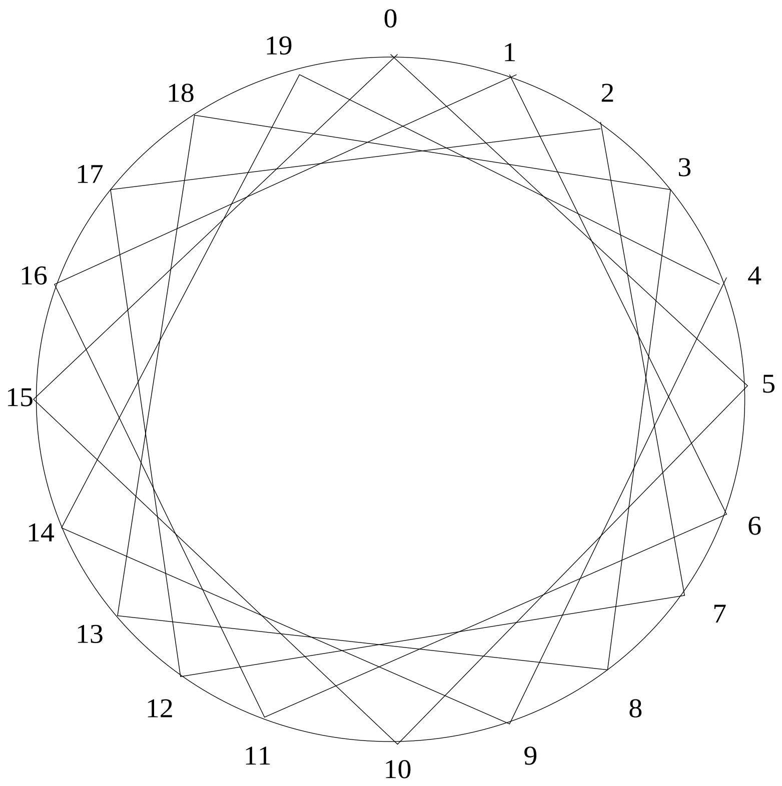


Figure 2.2: Distributed loop with $n = 20$ and $h = 5$

2.2 Terminology

In an MCDL of n nodes, every node i is connected to $(i + h)$, $(i - h)$, $(i + 1)$ and (i−1) (all in *modulo* n). These four link connections are distinguished as $[+h]$,$[-h]$,$[+1]$ and $[-1]$ links respectively. We also denote the number of $[+h]$, $[-h]$, $[+1]$ and $[-1]$ links of a path by w,x,y and z respectively. A traversal along the $[+h]$ or $[-h]$ links is called a *long hop* or simply *hop traversal* and the one along the $[+1]$ or $[-1]$ links is a *short hop* or simply *loop traversal*. Many times we are interested in paths in the network that contains either $[+h]$ or $[-h]$ links, and additionally either $[+1]$ or $[-1]$ links [MS95]. These paths are classified as $(+h, +1)$, $(+h, -1)$, $(-h, +1)$ and $(-h, -1)$ paths according to the link combination used. A path in $(+h, +1)$ link combination will be represented as $w[+h]+y[+1]$ indicating w number of $[+h]$ links and y number of $[+1]$ links. The length of such a path is $(w + y)$. One can similarly represent paths in $(+h, -1)$, $(-h, +1)$ and $(-h, -1)$ combinations as well. It is obvious that the order in which these long and short hops are used to cover the path is immaterial. For example in the $(+h, +1)$ link combination, a path of $2[+h] + 1[+1]$ can be traveled as either $[+h]$,$[+h]$,1$[+1]$ or $[+h]$,$[+1]$,$[+h]$ or $[+1]$,$[+h]$,$[+h]$ links.

Liu [Liu78] proposed a fault tolerant distributed system using $[+1]$ and $[-1]$ links and called

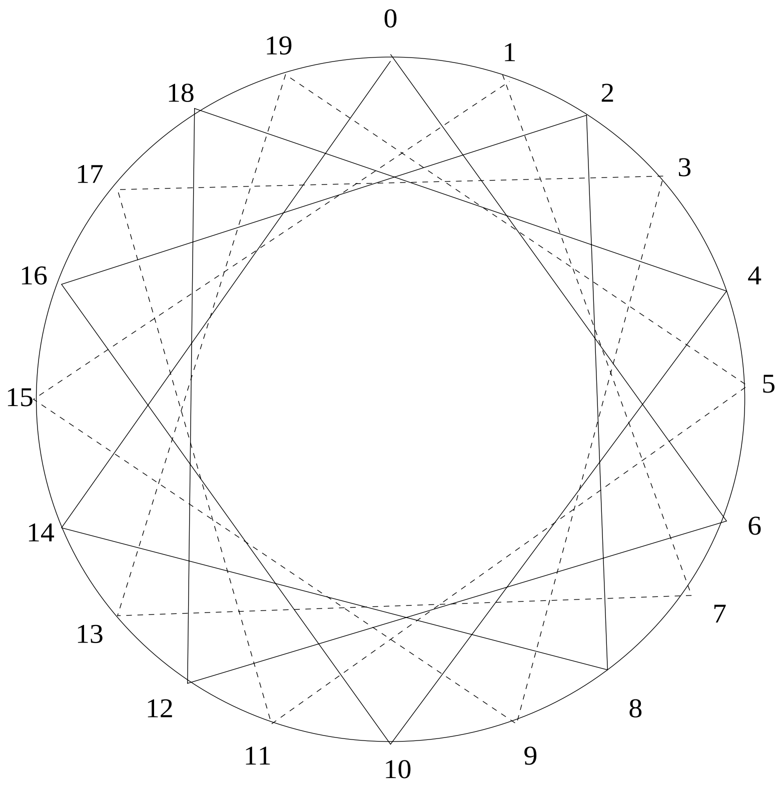


Figure 2.3: Distributed loop with $n = 20$ and $h = 6$

such a network a *distributed double loop computer network* (DDLCN). A *daisy chain* loop network that has [+1] and [−2] links was proposed by Grnanov *et al*, [GKG80] and was used in a fault tolerant system interconnection. Similarly, Raghavendra [RS86] proposed the *forward loop backward hop* (FLBH) network using [+1] and $[-h]$ links and *forward loop parity hop* (FLPH) network using [+1] and $[+h]$ or $[-h]$. The hop connections in FLPH are formed by connecting all odd nodes to $(i+h)$ $mod\, n$ and all even nodes to $(i-h)$ $mod\, n$. The FLPH with additionally [−1] links, odd h and even n is called *chordal ring* by Arden and Lee [AL81].

We use $G(n;+1,+h)$ to denote a loop network of n nodes with link connections [+1] and $[+h]$ only. Similarly $G(n;1,h)$ denotes the networks having all the four link types. Their diameters are represented by $dia'(n;+1,+h)$ and $dia(n,h)$ respectively. Thus $G(n;+1,+h)$ is the *directed version* and $G(n;1,h)$ is the *undirected version*. Networks $G(n;+1,-h)$, $G(n;-1,+h)$ and $G(n;-1,-h)$ with the other three link combinations are also possible. We prove an isomorphism among all these four in Section-3.1. Hence for the most part we focus on studying $G(n;+1,+h)$.

The shortest path, in the $G(n;+1,+h)$ from node l to node m is represented by $sp_{++}(l,m)$. Similar notation is used in networks with the other three link combinations. The overall shortest path from l to m in $G(n;1,h)$ is represented by $sp(l,m)$. Hence

$$sp(l,m) = \mathbf{Min}\ [\ sp_{++}(l,m), sp_{+-}(l,m), sp_{--}(l,m), sp_{-+}(l,m)\ ] \tag{2.1}$$

Average path length of the network, $ava'(n;+1,+h)$ and $ava(n,h)$ for the directed and undirected versions respectively are defined as [RS85]:

$$\begin{aligned} ava'(n;+1,+h) &= \frac{1}{n}\sum_{l=0}^{n-1}[\ \frac{1}{n-1}\sum_{m=0}^{n-1} sp_{++}(l,m)\ ], \qquad l \neq m \\ ava(n,h) &= \frac{1}{n}\sum_{l=0}^{n-1}[\ \frac{1}{n-1}\sum_{m=0}^{n-1} sp(l,m)\ ], \qquad l \neq m \end{aligned}$$

$\forall\ l,m\ \in \{0,1,\ldots n-1\}$. As the network is symmetric, they are

$$ava'(n;+1,+h) = \frac{1}{n-1}[\ \sum_{p=1}^{n-1} sp_{++}(0,p)\ ] \tag{2.2}$$

$$ava(n,h) = \frac{1}{n-1}[\ \sum_{p=1}^{n-1} sp(0,p)\ ] \tag{2.3}$$

$\forall\ p\ \in \{1,2,\ldots n-1\}$. We can now define the diameters of the directed and undirected network as:

$$\begin{aligned} dia'(n;+1,+h) &= \mathbf{Max}\ [\ sp_{++}(l,m)\ ] \\ dia(n,h) &= \mathbf{Max}\ [\ sp(l,m)\ ] \end{aligned}$$

$\forall\ l,m \in \{0,1\ldots(n-1)\}$ and $l \neq m$. Again in view of symmetry of the topology,

$$dia'(n;+1,+h) = \mathbf{Max}\ [\ sp_{++}(0,p)\ ] \tag{2.4}$$

$$dia(n,h) = \mathbf{Max}\ [\ sp(0,p)\ ] \tag{2.5}$$

$\forall\ p \in \{1,2,\ldots(n-1)\}$.

The diameter of a loop network is said to be *optimal* diameter if it is the smallest diameter among networks with all the possible hop sizes. We represent the optimal diameters $dia'(n)$ and $dia(n)$ for directed and undirected loops respectively. Hence they are,

$$dia'(n) = \mathbf{Min}_{\forall h}\ [\ dia'(n;+1,+h)\ ] \tag{2.6}$$

$$dia(n) = \mathbf{Min}_{\forall h}\ [\ dia(n,h)\ ] \tag{2.7}$$

Loops which achieve this diameter are called *optimal diameter loops*. Similarly *optimal average distance loops* are defined as loops which achieve the minimum average distance defined below:

$$ava'(n) = \mathbf{Min}_{\forall h}\ [\ ava'(n;+1,+h)\ ] \tag{2.8}$$

$$ava(n) = \mathbf{Min}_{\forall h}\ [\ ava(n,h)\ ] \tag{2.9}$$

We similarly define the average path length and diameter for the other three directed versions of MCDL.

It is interesting to note and will also be shown in Section 4.1 through the table 4.1 that loops which are optimal in diameter are not necessarily optimal in their average distance. In the following we present a survey on some studies related to shortest paths.

2.3 Graph properties of MCDL

One of the important issues in MCDL networks is computation of shortest path as part of routing of messages. Because, using a shortest path between nodes for message transmission has a direct impact on the delay and throughput of the network. Using these shortest path studies, we also propose to obtain diameter and average path length of MCDL topologies in the lines given in Section-2.2.

2.3.1 Shortest path studies

In [MS95] computation of shortest paths in MCDL topology has been reported in $\bigcirc(\delta)$ time, where δ is the diameter. We developed algorithms for finding shortest paths using number theoretic properties and linear congruences. The main feature of our algorithms is that shortest path is computed in time independent of the diameter, unlike in [MS95].

2.3.2 Diameter and average path length studies

A way of measuring the performance of interconnection networks is by the maximum distance (or diameter) over all pairs of nodes. A large diameter would contribute to large transmission delays in the network. Another related quantity is the *average distance* between all pairs of nodes in the network. A way to increase the connectivity and decrease the diameter (and average distance) in a ring network is by adding additional links to the loop network. These links must be added in a homogeneous way and must be as few as possible so that messages can be routed in a systematic way and the switching mechanism at every node can be implemented easily. In MCDL this is accomplished by hop connections. In an MCDL, for a given number of nodes n, one wants to find a smallest diameter loop as it reduces overall transfer delay of packets. But most of the time the exact value is unknown. In the following we present a survey for some bounds on this.

Arden and Lee in [AL81] initially proposed *chordal rings*, a variation of MCDL. Several studies on MCDL topology have been carried out. For example, routing has been studied by Arden and Lee [AL81], Raghavendra and and Silvester [RS85], Mukhopadhyaya and Sinha [MS95]. Fault tolerance has been studied by Masuyama and Ichimori [MI89], Peha and Tobagi [PT92], Hu and Hwang [HH92], Hwang and Wright [HW95]. The diameter of these networks were studied by Boesch and Wang [BW85], Hsu and Xu [DH90], Zerovnik and Pisanski [ZP93].

The problem of diameter optimality was studied by Foil *et al.* [FYAV87], Bermond and Tzvieli [BT91], Tzvieli [Tzv91]. Some of the recent studies in parallel processing applications for MCDL is reported in [Par95, HK94] in which reliability and fault tolerance were reported. An exhaustive survey on the MCDL networks is found in [RS86, BCH95].

For a directed ring network, the diameter is as big as $(n-1)$. For a DDLCN of $G(n;1)$ diameter is $\lfloor \frac{n}{2} \rfloor$ and average distance is $\frac{n}{4}$. The daisy chain which is $G(n;+1,-2)$ has diameter $\lfloor \frac{n}{3} \rfloor + 1$ and average distance of $\frac{n}{6}$. Moreover Wong and Coppersmith [WC74] have shown that for $G(t^2;+1,+t)$ has diameter exactly $2t-2$ (roughly $\sqrt{n}$, for large n) and average distance of $t-1$ (again roughly $\sqrt{n}$, for large n). It was also shown that

$$\begin{array}{rcccl} \lceil\sqrt{3n}\rceil - 2 & \leq & d'(n) & \leq & \lceil\sqrt{3n}\rceil - 1 \\ \frac{5}{9}\sqrt{3n} - 1 & \leq & a'(n) & \leq & \frac{5}{9}(\sqrt{3n} - 1) \end{array}$$

where

$$\begin{array}{rcl} d'(n) & = & \mathbf{Min}_{\forall h_1,\ h_2}[dia'(n;+h_1,+h_2)] \\ a'(n) & = & \mathbf{Min}_{\forall h_1,\ h_2}[ava'(n;+h_1,+h_2)] \end{array}$$

Loop networks which achieve the lower bound on diameter *i.e.*, $lb(n) = \lceil\sqrt{3n}\rceil - 2$ as given above are called *tight directed loops*. Not all $G(n;+1,,+h)$ can achieve this lower bound. Fiol *et al*. [FYAV87] has shown by an exhaustive search that for most values of n there exist networks such that $dia'(n) = d'(n)$ and the smallest n for which the equality fails is for $n = 450$. In this case $dia'(450) = 36 = dia(450;+1,+59)$ and $d'(450) = 35 = dia(450;+2,+185)$. There are infinite number of such loops that are not tight as given by Esque and Aguilo [EAF93]. They gave through computer search that for $1 \leq n \leq 30000$ there exist several values of n for which $dia'(n) - lb(n) = 3$. But for $1 \leq n \leq 75000$ there exist only three values for n that show $dia'(n) - lb(n) = 4$. They are $n = 53749$ $(dia'(53749) = dia'(53749;+1,+985) = 404)$, $n = 64729$ $(dia'(64729) = dia'(64729;+1,+394) = 443)$ and $n = 69283$ $(dia'(69283) = dia'(69283;+1,+1764) = 458)$. On the other hand it was not known until recently whether $lb(n)$ could be achieved for infinite values of n. Erdos and Hsu [EH92], and Esque *et al* [EAF93] reported a number of families of n in this direction. Thus the problem of optimizing $G(n;+1,+h)$ is not completely solved.

It was also shown by Fiol *et al.* that $dia'(n)$ does not increase monotonically. For example $dia'(20) = 7 = dia(20;+1,+4)$ while $dia'(21) = 6 = dia(21;+1,+9)$. Similarly, $ava'(n)$ and $ava'(n;+1,+h)$ also are not regular. As an example consider $ava'(20) = ava'(21;+1,+4) = 3.5$, but $ava'(21) = ava'(21;+1,+9) = 3.429$. It is also interesting to note that some loops that are optimal in diameter are not necessarily optimal for average distance. As an example $ava'(59) = ava'(59;+1,+8) = 6.5768$. But $dia'(59;+1,+8) = 13 > dia'(59) = dia'(59;+1,+25) = 12$.

Similar to the bound being given for the directed version, the undirected version $G(n;1,h)$

of MCDL is also studied. A more general form of this would be $G(n; h_1, h_2)$, where every node i is connected to $(i \pm h_1)$ and $(i \pm h_2)$ mod n. They are *connected* if $GCD(n, h_1, h_2) = 1$. Du [DH90] showed that

$$d(n) \quad \geq \quad \lceil \frac{\sqrt{2n-1}-1}{2} \rceil$$

where

$$d(n) \quad = \quad \mathbf{Min}_{\forall h_1, h_2} \; [dia(n; h_1, h_2)]$$

is the lower bound for the undirected version. The graphs $G(n; h_1, h_2)$ which achieve this $d(n)$ are called *tight undirected loops*. Tzvieli [Tzv91] has shown that $d(n)$ can be achieved if $h_1 = d(n)$ and $h_2 = d(n) + 1$. The problem of finding optimal diameter $dia(n)$ for $G(n; 1, h)$ is not completely solved. But it is known that $dia(2t^2 + 2t + 1, 2t + 1) = dia(2t^2 + 2t + 1) = t = d(2t^2 + 2t + 1)$ from Du. Tzvieli [Tzv91] conjectured that $dia(n)$ is at most $d(n) + 1$ and verified it up to $n = 8000000$. Du and Hsu [DH90] also classified n which are tight *i.e.*, that achieve $d(n)$. Bermond and Tzvieli [BT91] determined families of n for which $dia(n) = d(n)$. Tzvieli [Tzv91] also defined *suboptimal loops*, graphs which achieve $dia(n) = d(n) + 1$ and obtained a complete census of n that are also suboptimal. Using this census Hsu [DH90] gave that

$$dia(n) \quad < \quad (\frac{n}{2})^{\frac{1}{2}} - (\frac{n}{8})^{\frac{1}{4}} + 2.$$

A class of loop networks proposed by Arden and Lee [AL81] and called chordal rings have the diameter and average distance of $\bigcirc(\sqrt{n})$. They also gave a distributed algorithm for routing and determination of alternate route under node or link failure.

In our research though we also could not give a closed form expression for diameter and average distance, but we present an algorithm for both directed and undirected MCDL networks. We observed that these values are in accordance with those reported in [WC74, DH90, Tzv91, BT91, BW85, FYAV87].

2.4 Token based communication studies

Ring networks are popular interconnection networks in multiprocessor and parallel processing applications [Par95] and in distributed computing environment [PB79]. Loop networks in distributed computing were first introduced by Farmer and Newhall [FN69] in 1969 in distributed loop control for demand sharing of loop bandwidth. They used it for bursty and variable length message traffic communication. Their loop consists of a major loop and a number of minor loops. First theoretical study on loop networks was done by Pierce in 1974 on a loop he proposed [Pie72] *Pierce loop*. The *Pierce loop* is a three level loop: a national loop, several regional loops and many local loops, sufficient to serve a country. The distributed computer system project [FL72], which started in 1972 for research in operating system used a slotted

Table 2.1: Loop networks used in distributed computer environment.

Loop type	Loop Architecture	Loop Access Method
Farmer and Newhall	2-level hierarchical	Token Access
Pierce	Multi-level hierarchical	Slotted
Farber	Unidirectional	Slotted
Liu	Directed loop	Slotted
Cambridge Loop	A local loop	Multi-token
NSA Loop	Local loop	Multi-token
Karlsrushe Loop	Double loop fiber optic	Buffer insertion
IBM-2790 loop	Single loop	Time Slot
Weller loop	Single loop	Time Slot

loop with seven microcomputers on it. In this, they also realized reliability of the loop by the use of *node skipping* links. This loop is known as *daisy chained loop* [PB79]. Similar topologies are surveyed by Penny *et al.* and Harmen in [PB79, vA94]: *Spider loop, Distributed loop computer network (DLCN), Cambridge ring, National security agency loop (NSA loop), Double fiber distributed loop, IBM 2790 loop, Wellar loop, Serial Camac loop etc.* [PB79]

The *Spider loop* proposed by Fraser consists of a switching center providing local communications by means of several loops. The Distributed Loop Computer Network (DLCN) was designed in 1975 for interconnection of midi, mini and micro computers in a research project by Liu [FL72]. The control is completely distributed and no supervisor system was used. At University of Cambridge,UK a loop has been designed by Wilkes in the lines of Pierce loop for University resource sharing. This network is called the Cambridge Ring. In this loop, the sender is not allowed to send any more packets until the earlier packets are received and packets are not deleted at the destination. In 1973 National Security Agency reported on a design of a loop for resource sharing. In this slotted control technique was used. A double loop was also used for system reliability. In 1977-78 Heger developed a double fiber distributed loop for distributed computer control. Steward *et al.* designed IBM 2790 loop for connecting peripheral devices to a computer. They used polled mode for data transfer. Later Wellar Loop and Serial Camac Loops used the same IBM 2790 loop structure. All these loop systems are reported in the surveys of Penny and Harmen in [PB79, vA94]. We tabulate some of these loop architectures and their access methods in Table 2.1. The advantages of loops over store-and-forward networks [PB79], are: (i) messages are not completely read in, stored and forwarded. So buffer storage is not needed, thus reducing overall cost (partial exception is buffer insertion), (ii) routing algorithms are either absent or simple, thus fewer software resources, (iii) message delays are small and (iv) multiple destination addresses is easy to implement.

The term *local area networks* was first introduced by Clark *et al.* [CPR78] with ring topology. They used *token* for conflict-free channel access as contrasted with random access. Although token passing protocols have been designed for both ring and bus topologies [IEE82], our interest is in token ring LANs only. One of the important issues in token ring LANs is delay *vs* throughput analysis. Some of the earliest works on analysing token rings was by Konheim and Meister [KM72] and Cherukuri [CLL82]. In [Bux81] token rings are compared with other access protocols such as buffer insertion and CSMA/CD. A later analysis is from Sethi and Saydam [SS85] by taking into account constant message lengths (appropriate for voice messages), exponentially distributed message lengths (appropriate for data messages) and stochastic behavior of token rotation time. Later Bhuyan *et al*, found [BGY89] that under moderate and high traffic rates the performance of token ring LANs falls. Hence they proposed a *multiple rings* topology for providing high bandwidth, better delay-throughput performance even at high loads and also for reliable communication.

2.4.1 Principles of token control

The well known access protocols [PB79] based on ring topology are *token passing, slotted operation and register insertion operation.* In [CPR78], the authors elaborated the issues concerning topologies (such as ring, star and bus), protocols to be used, host interfaces, addressing of nodes and interconnection of different local area networks. The *Token passing* access mechanism is one of the control mechanisms described by them. In token passing, a unique bit pattern called *token* is passed from station to station to grant medium access rights. A ready station must wait for a free token before it can transfer. Once a free token is received by a station it converts the token to busy, transmits its packet and releases a new free token at the end. Various token passing schemes are possible depending on: (1) the time the free token is released, (2) whether the transmitted packet is removed from the ring by the receiving station or the sending station, and (3) whether all the waiting packets are transmitted with a free token (exhaustive service) or only a limited number of packets (non exhaustive service) are sent.

In the first scheme [Bux81], three approaches are examined regarding the time at which a free token is released. They are: *conservative approach* in which sender of a packet issues a new free token when it has completely erased its entire packet, *single-token approach* in which the sender of a packet does not issue a free token before it has received its token and *multiple token operation* in which a new free token is issued after the last bit of the packet has left the sending station. Bux [Bux81] analysed these three cases to compare the *mean packet transfer time* versus *mean throughput* and has shown that at higher ring latencies, the multiple token operation is better than the other two approaches.

In the second scheme, the approach of source removal of data entails fairness to access the medium among all the nodes, lost tokens can easily be recovered, destination nodes need not generate a separate acknowledgment packet and station latency can be only 1 bit to remove packets on the ring [QLL92]. But the disadvantage is bandwidth misutilisation. It is because the packet on average travels half way around the loop before reaching the destination and later around the second half back to the sender. This return journey does not convey any useful information and thus 50% of loop capacity is wasted. This reflects in bandwidth misutilisation. In destination removal strategy [QLL92], although packets need not travel beyond their destination, the disadvantages are that stations need a latency of at least equal to the length of destination address to remove packets from the ring. In fact destination removal is useful when media access allows more than one packet be in transit any time. Otherwise 50% of the wasted bandwidth can not be recovered. Also, analysis of destination removal seem to be complicated because of the dynamic location of the tokens and their associated bandwidths [QLL92, WK93]. However, Qu *et al.* [QLL92] proposed destination removal and Wee and Kamal [WK93] proposed a partial destination removal strategy. In partial destination removal strategy only the header of the received packet is transmitted back to the sender. But the station latency is kept at three bit delays in their media access protocol. They compared its performance with source removal. Both the works are from simulations and show that the destination removal gives better message delay and has a greater bandwidth availability. Cidon and Ofek [CO93] proposed Meta Ring in slotted operation with destination removal and claimed spatial reuse, the ability to provide concurrent transmission over distinct segments of the ring and obtained similar simulation results.

In the third scheme, Mukkamala *et al.* [MFMK93] simulated both the exhaustive and non exhaustive service policies. From their simulated models they concluded that mean delay of packet reduces in exhaustive service provided multiple rings (8 and above) are used. Similar gains due to 2 and 4 rings are negligible.

2.4.2 Problems with token ring at high speeds

A major problem in token passing is the initial waiting time to receive a free token. This initial waiting time could be appreciable especially in large rings. As a means of reducing this delay, token ring protocols with multiple tokens are proposed. This approach improves delay performance at light loads but decreases the maximum achievable throughput if the number tokens used is large. Increasing number of tokens beyond a limit results in rapid performance degradation at high load because of increased token overhead. The throughput of LANs is limited by the token hold time, packet lengths and speed of transmission [AK91]. Specifically, throughput depends on overhead time spent by a packet in waiting for a free token. More the overhead less will be the throughput. At high speeds the packet transmission time T will be comparatively small due to high data rate of the channels. Even for sufficiently large packet sizes it is possible that T becomes comparable to, or even less than the channel propagation delay τ. If α is defined as the ratio of the end-to-end propagation delay to the packet transmission time, then α could be close to 1 or even greater at high speeds. We can view α as the maximum number of packets that can be in transit in the transmission channel at any given time. Intuitively higher values of α appear to result in better performance. But not so. When α is high, the medium propagation delay is the dominant factor compared to the packet transmission time and hence the fraction of time spent in information transfer is very small. Much of the time is spent in the overhead such as waiting time for the free token to arrive. At higher values of α this waiting time overhead is also more because of larger propagation delays. This results in a rapid degradation of performance at high speeds with increasing α in many LAN schemes. But when α is small and less than 1, the packet transmission time is the dominating factor compared to the medium propagation delay and hence time spent on token waiting time overhead is comparatively small. Hence high channel utilization and lower delays may be achieved when α is small. For example, with a packet length of 2000 bits at 100 Mbps, the T is 20 μsec. Assuming propagation speed of 2×10^8 mtr/sec, in a LAN that spreads 5000 meters, the τ is 25 μsec. In this case α is 1.25. If the LAN spans 10000 meters, τ is 50 μsec and α is 2.5. When α is greater than 1, the bandwidth is also misutilised. For example, a ring of 10 km of circumference having $\tau = 50\mu$ sec, operating at 100 Mbps, the delay around the ring is 5000 bit intervals. Hence a host accessing the ring would have to transmit packets around 5000 bits per access in order to use the ring bandwidth efficiently. But storing 5000 bits in a cost effective memory capable of being read at 100 Mhz clock rate represents a significant hardware challenge. But if packets are smaller than 5000 bits α is greater than 1 and higher delays may result.

2.4.3 Analysis of token access

The token ring local area network was throughly studied and analysed by several researchers. In these analytic methods, exhaustive service policy [Liu78] and non-exhaustive service policy [Kuc79] have been studied. Sethi and Saydam [SS85] used a gated $M/G/1$ model with a non-exhaustive service policy. Our research was first motivated by these results of [SS85]. Their analytic results showed that the transmission delays faced by a packet is high under moderate and high traffic loads. This is due to higher token rotation time. It is due to these reasons that single ring networks are not well suited for distributed processing that requires high interprocess communication. Providing multiple rings and multiple tokens is an attractive alternative, because it achieves smaller token rotation times and better reliability. Multiple rings can also offer high bandwidth. Application of multiple rings in distributed processing are described in [BGY89]. In [BGY89], the authors also presented performance models for multiple rings with token and register insertion media accesses. The multiple ring network considered in [BGY89] is shown in Figure-2.4. The B rings are independent and operate asynchronously

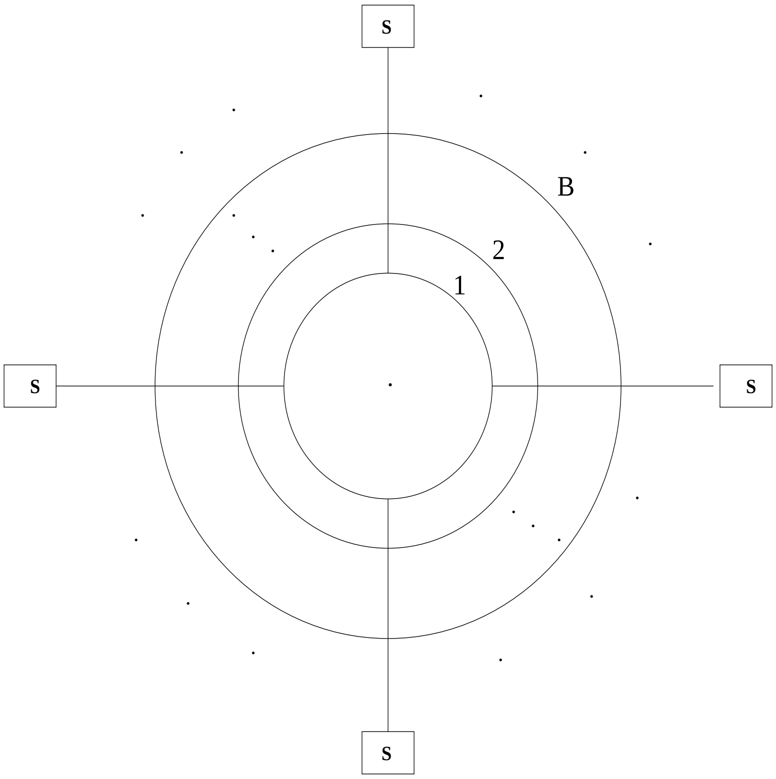


Figure 2.4: Multiple ring network

with a token in each of the ring. If a station has no packet to transmit the ring interface simply passes on the arriving tokens to the next station. When a station has a packet to transmit the ring interface simultaneously monitors all the rings and captures the first token arriving on any one of the rings. Then the token is converted into a connector and then immediately followed by the message packet. The token is thus transmitted immediately after the message packet. Each station is allowed to transmit only one packet at a time (non-exhaustive policy). During the transmission time at a station, if tokens arrive to this station from other rings they are passed on to the next station on the ring. The analytic results presented by them show that as the number of rings increases the delay performance is better than single ring networks. This is the second result that motivated our research on multiconnected loop topologies.

2.4.4 Waiting times in token access

In the token access control methods explained in 2.4.1, the waiting time has been studied and analysed using queueing theory techniques by several authors. We give below their results with the following notations.

M	Number of nodes.
r	Node-to-node propagation time including processing time of token.
λ	Average traffic arrival rate at a node (assumed Poisson arrival).
x	Packet transmission time (fixed packet lengths).
B	Number of parallel rings.

2.4.5 Waiting times in SRST

Consider M stations connected to a ring and assume that messages (of fixed length packets) arrive at each station in an independent Poisson process with an average arrival rate of λ. Assume that stations have infinite buffer size. Let the token take r bit times (constant) to move from one station to its next. A station which has a waiting message is called an *active station*. An active station waits for a token from the ring to transmit its packet. When a token is received if the station is active it changes the token to a connector data and appends its data to the ring. At the end of data transmission it releases the token. We call this *single ring single token (SRST) system*. We define token rotation time, as the elapsed time from the instant a station receives it till the instant the same station receives again. In one rotation time of the token, each station on the ring gets the opportunity to transmit a message (non exhaustive service). If all the M stations are active, the token rotation time is $M(x+r)$. However, not all M are active in a rotation. Let N be the random variable indicating the number of active stations, whose mean is $\overline{N}$. Let T be the random variable denoting the token rotation time and $\overline{T}$ denotes its mean. We get:

$$\begin{aligned} T &= N \cdot x + M \cdot r \qquad (2.10) \\ \overline{T} &= \overline{N} \cdot x + M \cdot r \end{aligned}$$

Thus the effective service time (T) at a station is generally distributed. Since arrivals are Poisson, each station becomes a standard $M/G/1$ queue with traffic intensity $\rho = \lambda \cdot \overline{T}$. Hence the waiting time in the station buffer is given by (C.6)

$$W = \frac{\rho(1+C_b^2)}{2\lambda(1-\rho)},$$

where C_b^2 is the squared coefficient of variation of service time and is given by

$$C_b^2 = \frac{\sigma_T^2}{(\overline{T})^2}$$

We find W here using the *probability generating functions p.g.f*, a method different from [SS85] in which the Laplace Transform methods are used. However, the same results can be seen from both the methods and this *p.g.f* method is simpler.

In the stable state of the network, ρ is < 1 and indicates the probability that a station is active (p_1). The probability that k stations are active (p_k) is given by

$$p_k = \left(\begin{array}{c} M \\ k \end{array} \right) \rho^k (1-\rho)^{M-k}$$

which is the standard binomial distribution. Thus N is binomially distributed. The $N(s)$, which is the p.g.f of N, $\overline{N}$ the mean and σ_N^2 the variance of N are (from Table C.1):

$$\begin{aligned} N(s) &= (\rho \cdot s + 1 - \rho)^M & (2.11) \\ \overline{N} &= M \cdot \rho \\ \sigma_N^2 &= M\rho(1-\rho) \end{aligned}$$

The $p.g.f$ of T, $T(s)$ is obtained from (2.10), (2.11) and Theorem-C.2 as:

$$T(s) = s^{Mr}(\rho s + 1 - \rho)^M$$

From (C.1) and (C.3) we get

$$\begin{aligned} \overline{T} &= M\rho x + Mr \\ \sigma_T^2 &= M\rho(1-\rho)x^2 \end{aligned}$$

Substituting the $\overline{T}$ and σ_T^2 in (C.6) and simplifying we get the waiting time of SRST system as

$$W = \frac{M}{2}\left[\frac{r}{1 - M \cdot \lambda \cdot x - M \cdot \lambda \cdot r} + x^2 \cdot \lambda\right] \qquad (2.12)$$

Assuming average path length to be $\frac{M}{2}$, the average delay of packets in SRST model is

$$D = W + x + \frac{M}{2} \cdot r \qquad (2.13)$$

The delay versus λ for different values of packet lengths are plotted in Figure-2.5. In Figure-2.5, we considered a bit rate of 10^5 bits/sec, $M = 50$ and $x = 100,\ 150,\ 200,\ 250$ and 300. The propagation and node processing times are considered to be 6 bits *i.e.*, $r = 6$. The λ and delay are represented in bit times.

2.4.6 Waiting times in MRMT

It can be seen from the Figure-2.5 that the delays are high not only for larger packet lengths but also for high traffic loads. It is due to the high communication latency. Providing high bandwidth through multiple channels (rings) is an attractive alternative. The multiple ring network is as shown in Figure-2.4. The channel access mechanism is as explained below. The B tokens in the B rings are independent and operate asynchronously. We call this *multiple ring multiple token (MRMT) system*. If the station has no packet to transmit, the ring interface simply passes the arriving token to the next station. An active station simultaneously monitors all the rings and captures the first arriving token on any of the B rings. The ring interface then transfers its data and immediately releases the token. Source removal of packets and non

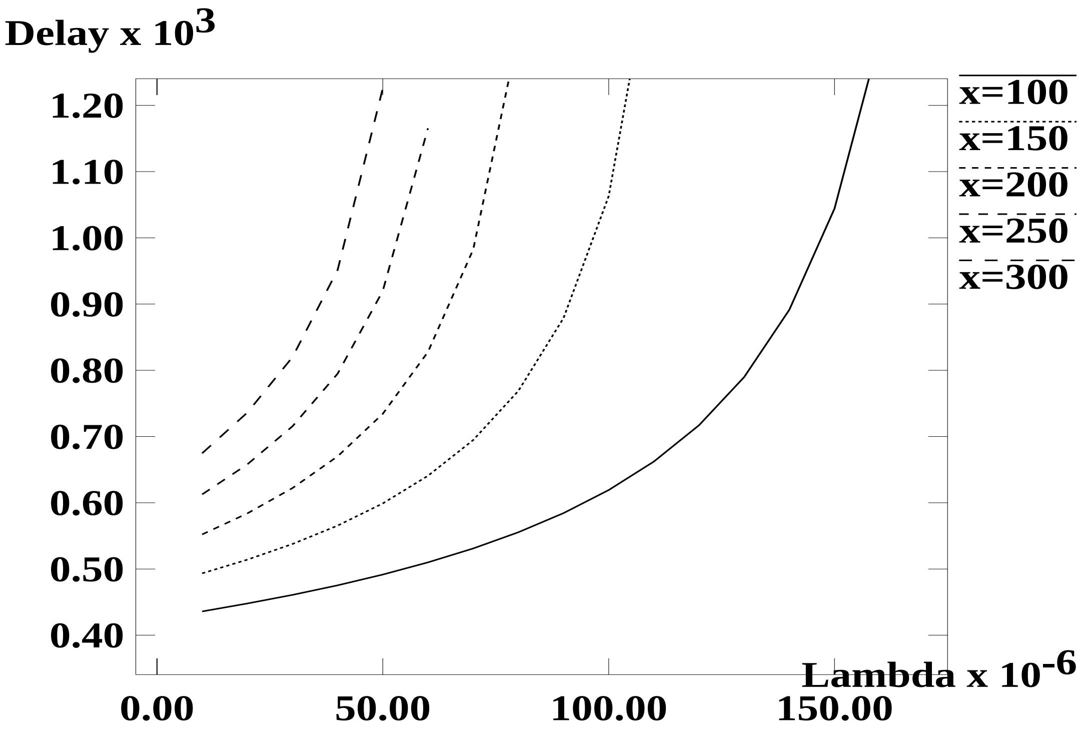


Figure 2.5: Packet delays in SRST with different packet lengths

exhaustive service is provided at each station. This implies that a station while transmitting a packet on a particular ring can not transmit another packet on any of the other rings until the ongoing transmission is completed. The tokens arriving at the station on the other rings when a station is transmitting a packet by holding one of the tokens, are passed on to the next station on the ring. However, it should be noted that a station can receive more than one packet simultaneously in order to avoid loss of packets. We define define T to be the time elapsed from the instant a station receives a free token on a particular ring until the next instant the same station receives another token on any of the B rings. Similarly, T_R is the residual life of the token, *i.e.*, the time for the token to arrive at the station measured from the instant a packet arrives. Thus $T_R = Min\ \{T_{R1}, T_{R2}, \ldots T_{RB}\}$, where $T_{R1}, T_{R2}, \ldots$ are the residual lives of the corresponding tokens on the B rings. We call T as the token *interarrival time*. T and T_R are random variables. Their mean is given below (for a proof see [BGY89]).

$$\overline{T} = \frac{M \cdot \rho \cdot x + M \cdot r}{B}$$

$$\overline{T_R} = \frac{M \cdot \rho \cdot x + M \cdot r}{B+1}$$

where

$$\rho \;=\; 1 - \lambda(P_{T<x}(x + \overline{T_R} + P_{T\geq x}(\overline{T} + \frac{B-1}{B} \cdot x))$$

$P_{T<x}$ Probability that token interarrival time is less than packet transfer time ;

$P_{T\geq x}$ Probability that token interarrival time is greater or equal to packet transfer time.

$P_{T<x}$ and $P_{T\geq x}$ are given by

$$P_{T<x} \;=\; 1 - (1 - \frac{x}{M \cdot \rho \cdot x + M \cdot r})^{B-1}$$

$$P_{T\geq x} \;=\; 1 - P_{T<x}$$

Using the above, the waiting time for packets is derived (see [BGY89] for details) as:

$$D \;=\; \frac{\overline{T_R}}{1 - \lambda(P_{T<x}(x + \overline{T_R}) + P_{T\geq x}(\overline{T} + \frac{B-1}{B} \cdot x))} + x + \frac{M}{2} \cdot r \tag{2.14}$$

We considered a bit rate of 10^5 bits/sec, $M = 50$ and $x = 100$ bits. The propagation time including the node processing time is considered to be of 6 bit times. Thus, $r = 6$. The delay versus λ is plotted for different values of B in Figure-2.6. The delay is expressed in bit times and λ is considered the number of packets that arrive in one bit time.

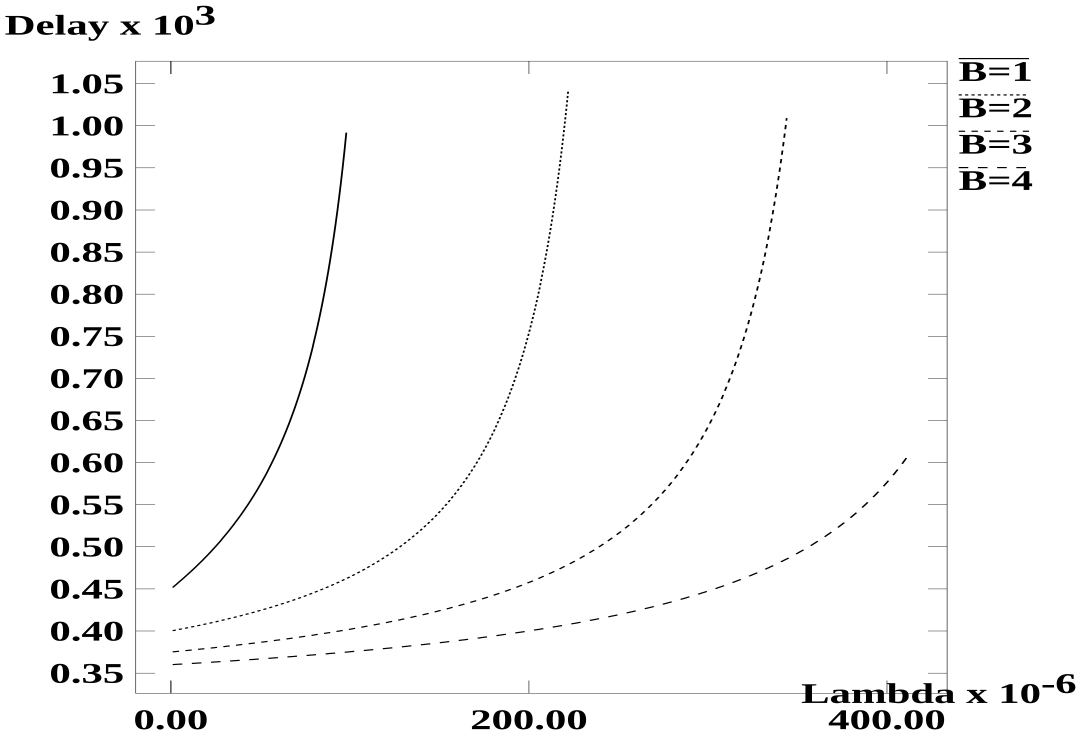


Figure 2.6: Packet delays in MRMT with varying number of rings (B=1..4)

2.4.7 Limitations of MRMT

Multiple ring networks can offer three implementations of protocols. They are
(i). Separate buffers with simultaneous transmissions,
(ii). Single buffer with simultaneous transmissions, and
(iii). Single buffer with single transmissions.
In [BGY89], single buffer with single transmissions is implemented. This scheme is simpler to implement than the other two models. As was seen in Figure-2.4, although an increase in number of rings decreases packet delay, it is not cost effective from the stand point of view of interface complexity and cabling. However, increase in number of rings makes the network more reliable. Network growth in terms of number of stations is also easy. Vasako and Rajasuman [VR94] performed simulation studies on the $MRMT$ model. In their simulation, they examined packet delay with 10,20 and 30 stations by increasing the number of rings from $1, 2, \ldots, 10, 20, 30$ respectively. However increasing number of rings beyond the number of stations is waste of resource. The simulations have shown that increasing the number of rings from 1 to 4 in case of 10 stations, from 1 to 8 in case of 20 stations and from 1 to 12 in case of 30 stations resulted in decrease of packet delays at a given load. But beyond them, the striking difference is that delay actually increases as the number of rings are increased. It is due to the limited arrival rate at the station's input causing a limitation and thus not being able to fully utilise the additional bandwidth provided by the large number of rings: for example more than 4 rings in the case of 10 stations. This also coincides with argument given by Bhuyan [BGY89] and a simple Lemma by Kamal in [Kam94]. Bhuyan points out that this is due to token *coalescence*. The phenomenon is due to the fact that the tokens move together at even moderate load and for very small packet sizes even if they are initially spaced at equal distance on all the rings. However Kamal [Kam94] proves that if packet lengths are very small compared to the round trip propagation time of the ring, it is likely that they move together (or cluster or coalesce) reducing the performance of multiple rings. Under these conditions of coalescence, the performance of MRMT becomes closer to the SRST model. Hence increasing number of rings more than some value does not contribute to the increased throughput and decreased delay. Although it is difficult to obtain an exact formula for optimal rings, Vasako and Rajasuman in [VR94] obtained a useful ratio of number of rings to the total number of stations. They obtained from simulations that the ratio to be in the range from 0.2 to 0.5.
Hence MRMT model is not well suited at all loads and packet lengths for distributed computing that requires high interprocess communication. Thus we wish to study *multiconnected distributed loop topology* (MCDL) in the token based commuinication applications.

2.5 Fault tolerance studies

The routing of messages is called a *static routing* if a predetermined fixed path between a given node pair in a network is used for routing. In a directed ring network messages travel in a fixed path and hence it is a static routing. Moreover failure of a link or node brings down the entire network. Therefore, a directed ring is not reliable unless all the links and nodes are reliable. In $multi - connected\ distributed\ loop$ networks discussed in Section-2.2, several alternate paths exist between a pair of nodes. Hence they can sustain node or link failures by routing via an alternate path. The routing of messages under such conditions is called *adaptive routing*. Such adaptive routing is possible in distributed loops in view of availability of multiple source-destination paths. We study in our work adaptive routing under faults (single node or link failures).

Fault tolerant routings were investigated in MCDL interconnection networks by several researchers in respect of node and edge faults [MS95, SLCH98, PR82, MI89]. Some of the studies are based on the reliability measurements under single and multiple faults [HW95, Ren80]. We base our study of fault tolerant routing in the lines of [MS95, SLCH98]. In [MS95] the existence of an alternative routing under node and edge is considered. In [SLCH98] some topological properties of multi connected loops are reported for single node and edge fault tolerance. In our work, we primarily consider two cases of knowledge of faults: *global knowledge of faults* and *neighbourhood knowledge of faults*. In global knowledge of faults, the occurrence of fault (node or edge fault) and its corresponding physical address in the network is available to all other nodes in the system. In neighbourhood knowledge of faults, only the nodes adjacent to the faulty component (node or edge) know the fault condition. We report in our work how efficiently a packet can be routed in a faulty MCDL under the two conditions. We also present cases for which there is an increases in the shortest path length.

2.6 Studies on deadlocks in wormhole switching

Many experimental and commercial parallel computers use *direct networks* for interconnection to achieve low latency and high bandwidth of interprocessor communication. The performance of multicomputer interconnection network depends on the *switching technique* and *routing algorithms* used. In our work we study the suitability of MCDL in multicomputer interconnection networks with wormhole switching and describe message routing and deadlock avoidance methods using virtual channels.

2.6.1 Direct network topologies

Scalability is an important factor in multiprocessor systems. Bus based systems are not scalable as the bus becomes the bottleneck when more processors are added. Direct network topologies are the most promising candidates for interconnection in multiprocessor and multicomputer architectures. They are point-to-point and scale well. In a direct network architecture each node has direct point-to-point connections to some number of other nodes called neighbouring nodes. Direct network topologies scale well which means as the number of nodes in the system increases the total communication bandwidth, memory bandwidth and processing capability of the system also increase. Each node is a small programmable computer with its own processor, local memory and other supporting devices. These nodes may be having different functional capabilities such as, vector processing, graphic capabilities and I/O processing *etc*,. A common component in a node is a *router* which handles communications among the nodes. For this reason direct networks are also m called *router based* networks. Usually two neighbouring nodes communicate via a pair of unidirectional channels in opposite direction. A bidirectional channel also may be used between adjacent nodes. Although the function of a router can be performed by local processor, a dedicated router is usually used allowing overlapped computation and communication with in each node. Nodes in the network communicate by passing messages whose size may vary depending on the application. For efficient use of network resources, messages are often divided into packets prior to transmission. A packet is the smallest unit of information that contains the destination address and sequencing information which are carried in the packet header. Neighbouring nodes may send packets to one another directly, while nodes that are not directly connected must rely on other nodes in the network to relay packets from source to destination. The router in the node handles such communication related tasks. The routers are connected by channels that define the topology of a direct network. A packet between two non-neighbouring nodes is forwarded by the router along a set of channels. The sequential list of channels is called the path and the number of channels in the path is the path length.

2.6.2 Characterization of direct networks

A direct network is characterized by three factors: topology, routing and switching. The fully connected topology would connect every node to every other node. No message would even have to pass through an intermediate node before reaching destination. This fully connected topology requires a router to have n links at each node for a network of n nodes. The cost of such a router is prohibitively high. Also the number of physical connections is limited by hardware constraints such as available pins and available wiring area. As a consequence, many other topologies have been proposed. In these topologies messages may have to traverse some intermediate nodes before reaching the destination and such topologies are called *direct network* topologies. Most commonly used direct network topologies are k-ary-n cube, mesh, tori and hypercube. Examples of parallel computers with interconnection using direct networks are

- *Cray T3E*: Bidirectional 3-D torus, 14-bit data links.
- *Cray T3D*: Bidirectional 3-D torus up to 1,024 nodes, 24-bit data links.
- *Intel Cavallino*: Bidirectional 3-D topology, 16-bit links.
- *SGI SPIDER*: Router with 20-bit links supporting regular and irregular topologies.
- *MIT M-Machine*: 3-D mesh.
- *MIT reliable router*: 2-D mesh, 23-bit links
- *Chaos router*: 2-D torus topology, 8-bit links.
- *Intel ipsc-2 Hypercube*: Binary hypercube, bit serial channel

The time required to move data between nodes is critical to system performance. A performance measure commonly used to evaluate direct network system s is *communication latency* defined as the sum of *start-up latency*, *network latency* and *blocking time* [NM93]. Start-up latency is the time required for a message framing/unframing, memory/buffer copying, validation and so on at both source and destination nodes. The startup latency mainly depends on the design of system software within the nodes and the interface between node and routers. The network latency is the elapsed time after the head of a message has entered the network at the source until the tail emerges from the network at the destination. If the network is contention free, given a source and destination node, the startup and network latencies are static values. The blocking time includes all possible delays encountered during the life time of a packet. These delays are mainly due to conflicts over the use of shared resources, for example, a busy channel or a full buffer. Blocking time reflects the dynamic behaviour of the network due to the passing of multiple messages whose traffic is unevenly distributed.

The unit of information exchange between the nodes is a *message*. For efficient and fair use of network resources, a message is often divided into fixed size packets prior to transmission. A packet carries the destination address and sequence number along with the data. The routing algorithms determine the path taken by a packet to reach its destination and indicates at each node indicates the next channel to be used. That channel may be selected among a set of possible choices. If all the potential channels are busy the packet is said to be *blocked* and the packet cannot advance further. Hence efficient routing is critical to the network performance. Routing algorithms that use minimal path (or shortest path) between source destination node pairs for message delivery are called *minimal routing algorithms*. A routing algorithm is *non-adaptive*, if it uses fixed paths between source destination node pairs. A routing algorithm is called *adaptive* if it provides multiple paths to use between source destination node pairs. The *minimal adaptive* routing algorithms do not impose any restriction on the choice of shortest path to be used in routing the messages.

When a packet header reaches an intermediate node, a *switching mechanism* determines how and when the router switch is set *i.e.*, the input channel is connected to the output channel selected by the routing algorithm. The switching mechanism determines allocation of network resources. In *circuit switching* all channels required by a message are reserved before starting transmission. In *packet switching* a packet is transmitted through a reserved channel but the next channel is not reserved until the packet releases the current channel. Obviously some buffer space is required to store the packet until the next channel is reserved. That buffer should be allocated before starting transmission. So buffer allocation is closely related to switching mechanism. The need to buffer a complete packet within the router makes it difficult to construct small, compact and fast routers. In *wormhole switching*, message packets are pipelined through the network. A message is broken into flits and is pipelined at the flit level. Thus at any instant of time a blocked packet occupies buffers in several routers. By this, the buffer requirements within a router are substantially reduced. In direct network topologies this method of *wormhole switching* has become quite popular in which the network latency is also independent of path length to be explained in Section 9.2.

Chapter 3

COMPUTING SHORTEST PATHS

We first show isomorphism among the networks $G(n;+1,+h)$, $G(n;+1,-h)$, $G(n;-1,+h)$ and $G(n;-1,-h)$ and apply it for finding shortest paths between node pairs in undirected multi-connected distributed loops. In using these link transformations, two approaches are proposed: one approach is to transform all the four into one link combination only (for example to $(+h,+1)$) and the other is to transform them into two link combinations (for example to $(+h,+1)$ and $(+h,-1)$). Thus the two methods are 'Method-1' and 'Method-2'. In both of them the number of long hops between a source/destination node pair in a link combination are given by two Theorems using Lemmas B.4, B.5, and B.6. Algorithms for these two methods are also compared with similar algorithms in literature.

3.1 Similarities among link combinations

We show here that the $(+h,-1)$,$(-h,+1)$ and $(-h,-1)$ link combinations can be transformed to $(+h,+1)$ link combination with suitable hop size and destination node modifications [CB98b]. Let w and y are the number of long hops and short hops from a node l to a node m in MCDL in $(+h+1)$ link combination, then the following equality holds good:

$$l + w \cdot h + y \quad = \quad m \ (mod\ n) \tag{3.1}$$

We can express similar relations between any source-destination pair in the other three link combinations also. Using these relations the following Lemmas are proved.

Lemma 3.1 *If $w[+h] + x[-h] + y[+1] + z[-1]$ is a path from l to m, then it is a path from 0 to p also, where $p = m - l\ (mod\ n)$.*

Proof: The path path from node l to node m relates l amd m from equation(3.1) as below:

$$\begin{aligned} m &= l + w \cdot h - x \cdot h + y - z \ (mod\ n) \\ m - l\ (mod\ n) &= w \cdot h - x \cdot h + y - z \ (mod\ n) \end{aligned}$$

Table 3.1: Transformations among link combinations

Link	Source-destination	Equivalent $(+h, +1)$	Equivalent source-destination
$(+h, -1)$	0 to p	$(+(n-h), +1)$	0 to $(n-p)$
$(-h, +1)$	0 to p	$(+(n-h), +1)$	0 to p
$(-h, -1)$	0 to p	$(+h, +1)$	0 to $(n-p)$

Hence $w[+h] + x[-h] + y[+1] + z[-1]$ is also a path from 0 to p, where $p = m - l \ (mod\ n)$. ♠

LEMMA 3.2 *If $w[+h]+z[-1]$ is a path from 0 to p in $(+h, -1)$ link combination, then $w[+(n-h)] + z[+1]$ is also a path from 0 to $(n-p)$ in $(+(n-h), +1)$link combination.*

Proof: *The path $w[+h] + z[-1]$ from 0 to p in $(+h, -1)$ link combination relates p as below from equation (3.1):*

$$\begin{aligned} p &= w \cdot h - z \ (mod\ n) \\ -p &= -w \cdot h + z \ (mod\ n) \\ n-p &= -w \cdot h + z \ (mod\ n) \\ &= -w \cdot h + z + w \cdot n \ (mod\ n) \\ &= w \cdot (n-h) + z \ (mod\ n) \end{aligned}$$

Hence from equation (3.1), $w[+(n-h)] + z[+1]$ is a path from 0 to $(n-p)$ in $(+(n-h), +1)$ link combination. ♠

LEMMA 3.3 *If $x[-h]+y[+1]$ is a path from 0 to p in $(-h, +1)$ link combination, then $x[+(n-h)] + y[+1]$ is also a path from 0 to p in $(+(n-h), +1)$ link combination.*

Proof: *The path $x[-h] + y[+1]$ from 0 to p in $(-h, +1)$ link combination is represented from equation (3.1) as:*

$$\begin{aligned} p &= -x \cdot h + y \ (mod\ n) \\ &= x \cdot (n-h) + y \ (mod\ n) \end{aligned}$$

Hence $x[+(n-h)] + y[+1]$ is a path from 0 to p in $(+(n-h), +1)$ link combination. ♠

LEMMA 3.4 *If $x[-h]+z[-1]$ is a path from 0 to p in $(-h, -1)$ link combination, then $x[+h] + z[+1]$ is also a path from 0 to $(n-p)$ in $(+h, +1)$ link combination.*

Proof: *The path $x[-h] + z[-1]$ from 0 to p in $(-h, -1)$ link combination is represented from equation (3.1) as:*

$$\begin{aligned} p &= -x \cdot h - z \ (mod\ n) \\ n-p &= x \cdot h + z \ (mod\ n) \end{aligned}$$

Hence $x[+h] + z[+1]$ is a path from 0 to $(n-p)$ in $(+h, +1)$ link combination. ♠

In summary, Table 3.1 gives the transformations. Thus finding shortest path in a $(+h, +1)$ network is equivalent to finding shortest path in $(+h, -1)$,$(-h, +1)$ or $(-h, -1)$ networks with appropriate tranformations of hop sizes and destination addresses.
So considering the $(+h, +1)$ link combination, we note that the number of $[+h]$ and $[+1]$ links to reach any node d form node 0 depends on the values of n and h. Thus, we need to study the relationship between n, h and the required number of $[+h]$ links to reach any node d. We use properties of modular arithmetic, specifically *Euclid's* algorithm and solutions of linear congruences which are explained in Appendix-B.

3.2 Computing long hops

We represent the greatest common divisor $GCD(n, h)$ by g, and when $g = 1$ the multiplicative inverse of h is represented by h^{-1}, so that $(h \cdot h^{-1})$ mod $n = 1$. In equation (3.1), $(w + y)$ is said to be the path length from node 0 to node p. For a shortest path length, we need to find w and y satisfying (3.1) subject to the condition that

$$Min\ [w + y]. \tag{3.2}$$

In a similar way we can formulate the finding of shortest path (and its length) in the other three link combinations also. The following Theorems and the methods to be given Sections 3.3 and 3.4 describe in finding shortest path in MCDL [CB98c].

THEOREM 3.1 *In a distributed loop network of n nodes if $g = 1$ then a node d, has a path $w[+h]$ from node 0 where $w = (d \cdot h^{-1})$ mod n.*

Proof: Let $d = 1$ and w be the number of $[+h]$ links needed to reach 1. Then

$$\begin{aligned} destination\ node &= source\ node + (number\ of\ [+h]\ links) \cdot h \\ 1 &= 0 + (w \cdot h)\ mod\ n \end{aligned}$$

This is a linear congruence equation to be solved for w.

Because $g = GCD(n,h) = 1$, as per the Property-B.3 of Appendix-B and Lemma B.4

$$w = (h^{-1})\ mod\ n,$$

where h^{-1} is the multiplicative inverse of h in modulo n.
So the number of $[+h]$ links needed to reach node 1 is h^{-1}. That is

$$(h^{-1} \cdot h)\ mod\ n = 1$$

Multiplying both sides by d, and applying Lemma B.2

$$(d \cdot h^{-1} \cdot h)\ mod\ n = d$$

This shows that any node d, has a path $(d \cdot h^{-1})$ mod n)$[+h]$ and hence it can be reached in $(d \cdot h^{-1})$ mod n number of $[+h]$ links, provided $GCD(n, h) = 1$. ♠

THEOREM 3.2 *In a distributed loop network of n nodes if $g > 1$, a node d of the form $d = i \cdot g$, $(i \geq 0)$ has a path $w[+h]$ from 0 where $w = (i \cdot h_1^{-1})$ mod n_1, $h_1 = \frac{h}{g}$ and $n_1 = \frac{n}{g}$.*

Proof: Let w be the number of $[+h]$ links needed to reach g, then

$$(w \cdot h) \; mod \; n \;\; = \;\; g$$

Using cancellation law given in Lemma B.3 and canceling g on both sides, we get

$$(w \cdot h_1) mod \; n_1 \;\; = \;\; 1$$

where $h_1 = \frac{h}{g}$ and $n_1 = \frac{n}{g}$. Because GCD$(n_1, h_1) = 1$, as per the Proposition B.3 Chapter B and Lemma B.4, we have

$$w \;\; = \;\; (h_1^{-1}) \; mod \; n_1$$

That means, to reach node g we need h_1^{-1} number of $[+h]$ links. So

$$(h_1^{-1} \cdot h_1) \; mod \; n_1 \;\; = \;\; 1$$

Multiplying both sides by ig and applying Lemma B.3

$$\begin{aligned} ((i \cdot h_1^{-1}) \cdot h_1 \cdot g) \; mod \; n_1 &= i \cdot g \\ ((i \cdot h_1^{-1}) \cdot h) \; mod \; n_1 &= i \cdot g \end{aligned}$$

Thus, the number of $[+h]$ links from 0 to $i \cdot g$ is $(i \cdot h_1^{-1})$ mod n_1. ♠

3.3 Method-1

A prelimianry version of this appeared in [CB98b]. In the method, we take node 0 as source and node p as destination. We also assume a $(+h, +1)$ network. It is proved in [MS95] through Lemma 4 that the number of $[+h]$ links in a shortest path from 0 to p is at least $L = \lfloor \frac{p}{h} \rfloor$ in $(+h, +1)$ link combination. The remaining distance $d = p \; (mod \; h)$. Clearly, this remaining distance is less than h. It remains to find a shortest path to cover this remaining distance. We differ from [MS95] in finding the number of $[+h]$ links to cover this remaining distance. Firstly, as given in Section 3.1 we map the problem of finding shortest path for this remaining distance to that of finding the same from node 0 to node d. Let $w_d[+h] + y_d[+1]$ be the shortest path from 0 to d. We can split this path in to two paths: $(w_d[+h] + 0[+1])$ and $(0[+h] + y_d[+1])$. It is obvious that the order in which these w_d and y_d are used to cover the distance is immaterial. The first path has no $[+1]$ links. So the w_d number of $[+h]$ links in this path lead to a node which is multiple of g as per Theorem-3.2. Let us say that this node is $i \cdot g$. The remaining distance $(d - i \cdot g)$ has to be covered in second path by $[+1]$ links. Thus the total path length to d is

$$w_d + y_d \;\; = \;\; (i \cdot h_1^{-1}) \; mod \; n_1 + (d - i \cdot g)$$

Table 3.2: Algorithm complexities in the four link combinations

Link	Equivalent $(+h, +1)$	Complexity
$(+h, +1)$	- - -	$\frac{h}{g} + log(h)$
$(+h, -1)$	$(+(n-h), +1)$	$\frac{n-h}{g} + log(n-h)$
$(-h, +1)$	$(+(n-h), +1)$	$\frac{n-h}{g} + log(n-h)$
$(-h, -1)$	$(+h, +1)$	$\frac{h}{g} + log(h)$

in which we need to find i such that the above is minimum for $0 \leq i \leq \lfloor \frac{d}{g} \rfloor$. The value of i for which it is minimum, gives w_d and y_d. An algorithm to compute i is given in Algorithm 3.1.

ALGORITHM 3.1 *PATH(n,h,l,m) //To find shortest path of* $w[+h] + y[+1]$ *for a distributed loop of* n *nodes,* h *hop size, from node* l *to* m*//*

Step 1. $L = \lfloor \frac{(m-l)\ mod\ n}{h} \rfloor$

$d = ((m\text{-}l)\ mod\ n)\ mod\ h$

Step 2. $g = GCD(n,h)$

$h_1 = \frac{h}{g},\ n_1 = \frac{n}{g}$

$h_1^{-1} = MULTINV(n_1, h_1)$

Step 3. $w_d = 0 \quad y_d = d$

LOOP $i = 0$ *to* $\lfloor \frac{d}{g} \rfloor$

$w_i = (i * h_1^{-1})\ mod\ n_1$

$y_i = d - i * g$

IF $(w_i + y_i) < (w_d + y_d)$

$w_d = w_i \quad y_d = y_i$

END IF

END Loop

Step 4. $w = w_d + L$

$y = y_d$

Return (w,y)

End PATH. ♠

The LOOP in step 3 executes a maximum of h/g times, when d is $h-1$. The GCD and h_1^{-1} in step 2 can be computed in $\bigcirc(\log h)$ and in $\bigcirc(\log(\frac{h}{g}))$ time respectively as given in Appendix-B.1. Hence the algorithm complexity is $\bigcirc(\frac{h}{g} + \log h + \log(\frac{h}{g})) = \bigcirc(\frac{h}{g} + \log(h))$.

Now using the transformations in Table 3.1 the shortest paths in $(+h, -1)$, $(-h, +1)$ and $(-h, -1)$ can be computed as in Algorithm 3.1. The complexities are tabulated in Table 3.2. From the Table 3.2 the complexity of finding shortest path $sp(l, m)$ in $G(n; 1, h)$ is $\bigcirc(Max\ [\frac{h}{g},\ \frac{(n-h)}{g}] + Max\ [\log(h),\ \log(n-h)]) = \bigcirc(\frac{n}{g} + \log h + \log n) = \bigcirc(\frac{n}{g} + log\ n)$ as $h \ll n$.

3.4 Method-2

In this method [CB98c], first we find shortest path in $(+h, +1)$ link combination as in Section 3.3. Then we apply the link transformation for $(-h, -1)$ to $(+h, +1)$ (refer Table 3.1) and find shortest path similarly in $\bigcirc(\frac{h}{g} + log(h))$ time. We *do not* do other link transformations for the $(+h, -1)$ and $(-h, +1)$ to $(+h, +1)$. Instead we proceed as below for these combinations. But first let us prove a lemma.

LEMMA 3.5 *If $w[+h] + z[-1]$ is a path from 0 to p in $(+h, -1)$ link combination, then $w[-h] + z[+1]$ is a path from 0 to $(n-p)$ in $(-h, +1)$ link combination.*
Proof: The path $w[+h] + z[-1]$ from 0 to p in $(+h, -1)$ link combination is represented as:

$$\begin{aligned} p &= w \cdot h - z \ (mod\ n) \\ -p &= -w \cdot h + z \ (mod\ n) \\ (n-p) &= -w \cdot h + z \ (mod\ n) \end{aligned}$$

Hence, $w[-h] + z[+1]$ is a path from 0 to $(n-p)$ in $(-h, +1)$ link combination. ♠

This Lemma 3.5 helps to study only $(+h, -1)$ link combination instead of both the $(+h, -1)$ and $(-h, +1)$.
Now consider $(+h, -1)$ link combination to find shortest path from node 0 to node p. Let $w[+h] + z[-1]$ be the shortest path in this link combination. Path from 0 to p may have only $(n-p)$ number of $[-1]$ links and no $[+h]$ links. Hence the path is $(n-p)[-1]$. We call this the *trivial path*. When $w \neq 0$ we divide the shortest path into two paths: $w[+h] + 0[-1]$ and $0[+h] + z[-1]$. The first path which has only $[+h]$ links should reach a node which is a multiple of g. Let this node be N_g. This node N_g should be greater than p. But if N_g were less than p *ie*, $0 \leq N_g < p$, then in the second path, $0[+h] + z[-1]$, it needs more than the $(n-p)$ number of $[-1]$ links resulting in overall path length larger than the trivial case. Thus $p \leq N_g \leq (n-1)$. In order to further localize N_g between p and $(n-1)$, we consider two sub-ranges in this. They are *lower sub-range*: $p \leq N_g \leq \lceil \frac{p}{h} \rceil \cdot h$ and *upper sub-range*: $\lceil \frac{p}{h} \rceil \cdot h < N_g \leq (n-1)$. The number of $[-1]$ links and also $[+h]$ links to node p, from any node in the upper sub-range is greater than that from $\lceil \frac{p}{h} \rceil \cdot h$. So we ignore this upper sub-range. Now we consider only the lower sub-range. Recalling that N_g is multiple of g, the node N_g can be generalized in this lower sub-range as

$$N_g \quad = \quad (\lceil \frac{p}{g} \rceil + i) \ g$$

The value of i will be varying from 0 to $(\lceil \frac{p}{h} \rceil \cdot h_1 - \lceil \frac{p}{g} \rceil)$. For each i, the number of $[+h]$ links from 0 to N_g can be obtained from Theorem-3.2 and the $[-1]$ links are $N_g - p$. So

$$(w+z) \quad = \quad (\frac{N_g}{g} \cdot h_1^{-1}) \ mod\ n_1 + (N_g - p) \ mod\ n$$

The case when $z = 0$ is included in the above for $i = 0$ and $p \ mod \ g = 0$. The shortest path's length is obtained by finding the minimum of the above and the trivial path length $(n - p)$. So the shortest path length is

$$MIN \ [\ (n-p), \ (\frac{N_g}{g} \cdot h_1^{-1}) \ mod \ n_1 \ + (N_g - p) \ mod \ n \ ]$$

for $i = 0$ to $(\lceil \frac{p}{h} \rceil \cdot h_1 - \lceil \frac{p}{g} \rceil)$. A detailed algorithm is given in Algorithm 3.2. From the algorithm, the LOOP in step 2 executes a maximum of $\frac{h}{g}$ times. The GCD and h_1^{-1} in step 1 can be computed in $\bigcirc(\log(h))$ and in $\bigcirc(\log(\frac{h}{g}))$ time respectively as given Appendix-B.1. Hence the algorithm complexity is $\bigcirc(\frac{h}{g} + \log(h) + \log(\frac{h}{g})) = \bigcirc(\frac{h}{g} + \log(h))$.

ALGORITHM 3.2 *PATH(n,h,l,m) //To find shortest path of $w[+h] + z[-1]$ for a distributed loop of n nodes, h hop size, from node l to m//*

Step 1. $p = (m - l) \ mod \ n$
$g = GCD(n, h)$
$h_1 = \frac{h}{g}$ $\quad n_1 = \frac{n}{g}$
$h_1^{-1} = MULTINV(n_1, h_1)$

Step 2. $w = 0 \ , \ z = n - p$
// k is the upper bound for i //
$k = \lceil \frac{p}{h} \rceil \cdot h_1 - \lceil \frac{p}{g} \rceil$
LOOP $i = 0$ *to* k
  $N_g = (\lceil \frac{p}{g} \rceil + i)g$
  $w_i = (\frac{N_g}{g} \cdot h_1^{-1}) \ mod \ n_1$
  $z_i = N_g - p$
  IF $(w_i + z_i) < (w + z)$
    $w = w_i \ , \ z = z_i$
  END IF
END Loop

Step 3. *Return* (w, z)

End PATH. ♠

For $(-h, +1)$ link combination, we apply Lemma 3.5 and use algorithm 3.2 to find shortest path in $\bigcirc(\frac{h}{g} + log(h))$ time. So in all the four link combinations $sp(l, m)$ can found in $\bigcirc(\frac{h}{g} + \log(h))$ time, which is an improvement not only over [MS95] but also over the Method-1 given in Section 3.3.

In [MS95] the authors had obtained the shortest path in $\bigcirc(\delta)$ time (δ is diameter). But there exist no closed form for diameter (refer Section 2.3.2). Our algorithm on the other hand computes shortest path in time $\bigcirc(\frac{h}{g} + \log(h))$ that is indepenent of diameter. Further, our algorithm clearly performs better in cases of graphs with large diameter, but relatively small value of h.

Chapter 4

DIAMETER AND AVERAGE PATH LENGTHS

We present here optimal MCDL topologies, their characteristics and tabulate some optimal multi-connected loops. We also compare the MCDL with other topologies such as k-ary r-cube, mesh and torus.

4.1 Computation of diameter

In Chapter 3 we saw that the shortest path from 0 to p was computed from finding the minimum of the minimum in each of the four link combinations, repesented by equation (2.1). We elaborate the shortest path in each of the four link combinations below using the method given in Section 3.4.

After the minimum number of long hops are traveled, the remaining distance $d = p\ (mod\ h)$ is represented in Figure 4.1(a) for $(+h, +1)$ link combination. Using the link transformation given in Table 3.1 for the $(-h, -1)$ link combination, the destination becomes $p_1 = (n - p)$ and accordingly after $L_1 = \lfloor \frac{p_1}{h} \rfloor$ long hop traversals the remaining distance d_1 is $p_1\ (mod\ h)$ as shown in Figure 4.1(b). Let $h_1 = \frac{h}{g}$ and $n_1 = \frac{n}{g}$, then from Algorithm-3.1 we have:

$$\begin{aligned} sp_{++}(0,p) &= L + \overset{Min}{\forall i} \left[(i \cdot h_1^{-1})\ mod\ n_1 + d - i \cdot g \right], \quad i = 0\ to\ \lfloor \frac{d}{g} \rfloor \\ sp_{--}(0,p) &= L_1 + \overset{Min}{\forall j} \left[(j \cdot h_1^{-1})\ mod\ n_1 + d_1 - j \cdot g \right], \quad j = 0\ to\ \lfloor \frac{d_1}{g} \rfloor \end{aligned}$$

Similarly in the case of $(+h, -1)$ for $sp_{+-}(0,p)$, the d is given in Figure 4.1(c). Again using link transformations for sp_{-+} from $(-h, +1)$ to $(+h, -1)$ using Lemma-3.5, the d_1 is shown in

Figure 4.1(d). Thus the $sp_{+-}(0,p)$ and $sp_{-+}(0,p)$ are:

$$\begin{aligned} sp_{+-}(0,p) &= \overset{Min}{\forall k}\left[(n-p),\ L+(k\cdot h_1^{-1})\ mod\ n_1+(k\cdot g-d)\right],\ k=\lceil\frac{d}{g}\rceil\ to\ (h_1-1) \\ sp_{-+}(0,p) &= \overset{Min}{\forall l}\left[(n-p_1),\ L_1+(l\cdot h_1^{-1})\ mod\ n_1+(l\cdot g-d_1)\right],\ l=\lceil\frac{d_1}{g}\rceil\ to\ (h_1-1) \end{aligned}$$

Combining the above four we write

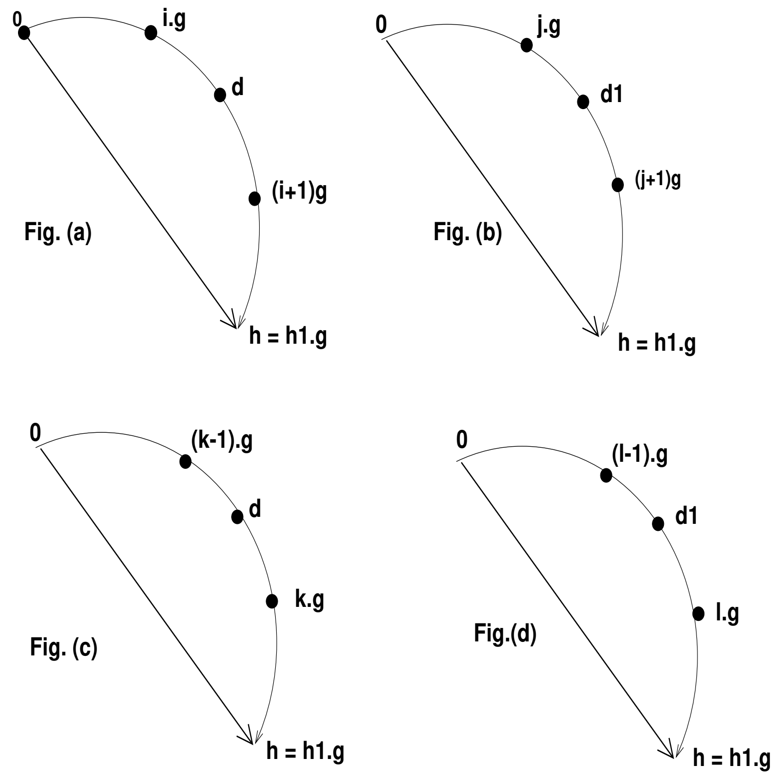


Figure 4.1: Unification of the four link combinations

$$sp(0,p) = \underset{i=0\ldots h_1-1}{\mathbf{Min}} \begin{pmatrix} L+(i\cdot h_1^{-1})\ mod\ n_1+d-i\cdot g & i\le\lfloor\frac{d}{g}\rfloor \\ (n-p),\ L+(i\cdot h_1^{-1})\ mod\ n_1+i\cdot g-d & i\ge\lceil\frac{d}{g}\rceil \\ L_1+(i\cdot h_1^{-1})\ mod\ n_1+d_1-i.g & i\le\lfloor\frac{d_1}{g}\rfloor \\ (n-p_1),\ L_1+(i\cdot h_1^{-1})\ mod\ n_1+i\cdot g-d_1 & i\ge\lceil\frac{d_1}{g}\rceil \end{pmatrix}$$

Using the above combined equation of $sp(0,p)$, equations (2.5) and (2.3), we obtain diameter and average distance for undirected networks. Similarly, the diameter and average distances for directed networks can also be obtained from (2.4) and (2.2). We give some results in Table 4.1 listing optimal loops (in respect of diameter and average distance) from (2.7) and (2.9) for undirected network. These results tally with those reported in [Tzv91, FYAV87, BW85]. Optimal diameter and optimal average distnace *vs.* number of nodes are plotted in Figures 4.2 and 4.3 respectively.

Table 4.1: Optimal multi-connected loops and their diameter/average path lengths

n	$dia(n)$	$ava(n)$	h for $dia(n)$	h for $ava(n)$
20	3	2.157895	8	8
21	3	2.200000	6,8	6,8
22	3	2.238095	6	6
23	3	2.272728	5,9	5,9
24	4	2.391304	4,5,6,7,9,10	10
25	3	2.333333	7	7
26	4	2.400001	4,6,7,10,11	10
40	5	3.051233	6,7,9,11,12, 15,16,17	6
41	4	3.000000	9	9
42	5	3.048782	9,12,16,18	16
43	5	3.095238	5,8,9,12,16, 17,18,19	12,18
65	6	3.812500	10,12,14,18,25,27	10,18

From the Table 4.1, it can be stated that *the diameter and average path length do not increase monotonically either with n or h.* For example as n increases, say from 23 to 26 the corresponding optimal diameters respectively are 3, 4, 3, *and* 4. The non-monotonicity can be observed at these values of n. Similarly the optimal diameter is attainable at $h = 5$ for $n = 23$. But for $n = 24$ it is attainable at a smaller hop size *ie.*, $h = 4$. Similar observations are found at other values of n also (refer Table 4.1). In the case of optimal average distance also the non-monotonicity can be observed as n changes from 24 to 25 and again when n changes from 40 to 41 *etc*,.

It is aslo possible to say from Table 4.1 that, *loops which are optimal in diameter for a hop size are not necessarily optimal in average path length for the same value of hop size. But the converse is true.* For example $ava(65) = 3.8125$ at $h = 10$ and 18. But $dia(65) = 6$ at $h = 10, 12, 14, 18, 25$ and 27.

We observe from Table 4.1 that *the non-monotonicity is because of the presence of mod operator in the algorithms of shortest path, diameter and average path lengths.* In fact mathematically the *mod* operator results in some kind of *scrambling* operation. It is used in *multiplicative congruential random number generator* [Deo97] used most commonly in generating a random number. Typically random numbers are generated using the following recursion that contains the *mod* operator as:

$$x_i = x_{i-1} \cdot a \ (mod\ m) \qquad i \geq 1 \tag{4.1}$$

where x_i is the i^{th} random and x_0 is the *seed* value. The a and m are constants. If $x_0 = 3,\ a = 7$ and $m = 15$ then $x_i = 6, 12, 9, 4$ for $= 1 \ldots 4$ respectively. Thus as i increases, the successive $x_i^{\prime}s$ are non-monotonic and random. If $x_0 = odd,\ m = 2^r\ (r > 2)$ and $a = 8 \cdot k \pm 3$ ($k =$ is a non-negative integer), the periodicity of random number is maximum and is equal to 2^{r-2}. Now let us observe in this context, the equation for computing the number of long hops in Step 3 of Algorithm-3.1 and 3.2. We reproduce below:

$$w_i \quad = \quad i \cdot h_1^{-1}\ (mod\ n_1) \tag{4.2}$$

where w_i is the number of long hops to node i from node 0, n is number of nodes and h is the hop size. Recall that $g = GCD(n, h),\ n_1 = \frac{n}{g},\ h_1 = \frac{h}{g}$. Comparing the two equations (4.1) and (4.2), we say that w_i is a random number and non-monotonic with respect to i the destination node address, given that h_1^{-1} is a constant for a network of n nodes (constant). So also are the $w_i + y_i$, the shortest path length, the diameter and average path length that are also non-monotonic with respect to i. Thus the optimal results for path lengths and diameter will be too non-monotonic.

4.2 Comparisons with other topologies

We compare the multi-connected loop topology with other topologies [CB98e] such as k-ary r-cube (with wraparound connections), mesh and tori in respect of diameter. A k-ary r-cube having wraparound connections contains k^r nodes in r dimensions. If $k = 1$, k-ary r-cube is a ring with k nodes. If $k = 2$ and $r = 3$ it is a hypercube with wraparound connections. If $k = 2$, then every node has r neighbours one in each dimension. If $k > 2$, then every node has $2r$ neighbors, two in each dimension. A 2-dimensional mesh has $k \times k$ number of nodes, k nodes in a dimension . Each node has 2 to 4 neighbours depending on its location. A torus is a $k \times k$ mesh with wraparound connections such that a node in a 2-dimensional torus has 4 neighbours. The degree of a network is the maximum number of neighbours that any node has. It is four for mesh and tori and $2 \cdot r$ for k-ary r-cube. The properties of these three topologies are tabulated in Table 4.2. In Table-4.3 the diameter comparisons are tabulated.

The degree and dimensions of loops are only 4 and 3 respectively, while these are dependent on r for k-ary r-cube. Hence it is easy to construct a multi-connected loop than a k-ary n-cube. For example a 4096 node in 8-ary 4-cube has degree 8, links 16,384 in 4 dimensions with a diameter of 16. With the same number of nodes, we can construct a multi-connected loop having half the degree and links (*ie* 4 and 8,192). But such a loop has an optimal diameter of 45 which is almost 2.8 times of k-ary r-cube. Thus though the constructional costs of loops are cheaper in comparison to meshes and tori, there may not be much gain in constructional

Table 4.2: Comparison of network characteristics with k-ary r-cube, mesh and tori

	k-ary r-cube	Mesh	Tori	Distributed Loop
N	k^r	$k \times k$	$k \times k$	N
Degree	$2 \cdot r$	4	4	4
Links	$r \cdot k^r$	$2 \cdot N - 2 \cdot \sqrt{N}$	$2 \cdot N$	$2 \cdot N$
Symmetric	Yes	No	Yes	Yes
Diameter	$r \cdot \lfloor \frac{k}{2} \rfloor$	$2 \cdot (\sqrt{N} - 1)$	$2 \cdot \lfloor \frac{\sqrt{N}}{2} \rfloor$	No closed form

Table 4.3: Diameter comparison of loops with k-ary r-cube,mesh and tori

Nodes	k-ary 4-cube	Mesh	Tori	Distributed Loop
16	4 in 2-ary 4-cube	6	4	dia(16)=3 at h=4 and 6
81	6 in 3-ary 4-cube	16	9	dia(81)=6 at h=24
256	8 in 4-ary 4-cube	30	16	dia(256)=11 at h= 92
625	10 in 5-ary 4-cube	48	25	dia(625)=18 at h=32,43,116,146,185,194, 218,242,255,264,293 and 297
1296	12 in 6-ary 4-cube	70	36	dia(1296)=26 at h=114,138,348,364, 411,452,510 and 621
2401	14 in 7-ary 4-cube	96	49	dia(2401)=35 at h=64,307,352,386,524 581,1163
4096	16 in 8-ary 4-cube	126	64	dia(4096)= 45 at h=90

costs. We find a good degree of advantage in terms of the diameter and thus message transfer delays as the delays depend on the diameter of a network. It is clear from the Table 4.3 that as n increases the diameters of optimal loops are smaller than that of mesh and tori having the same number of nodes.

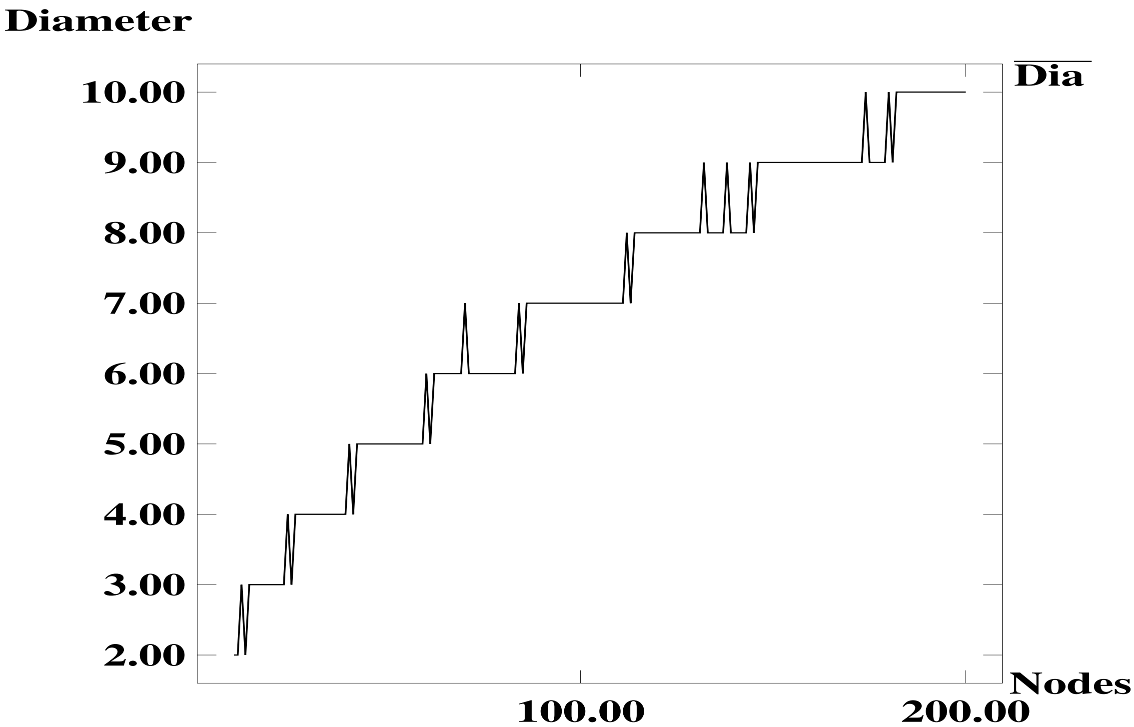


Figure 4.2: Diameter versus number of nodes in optimal loops

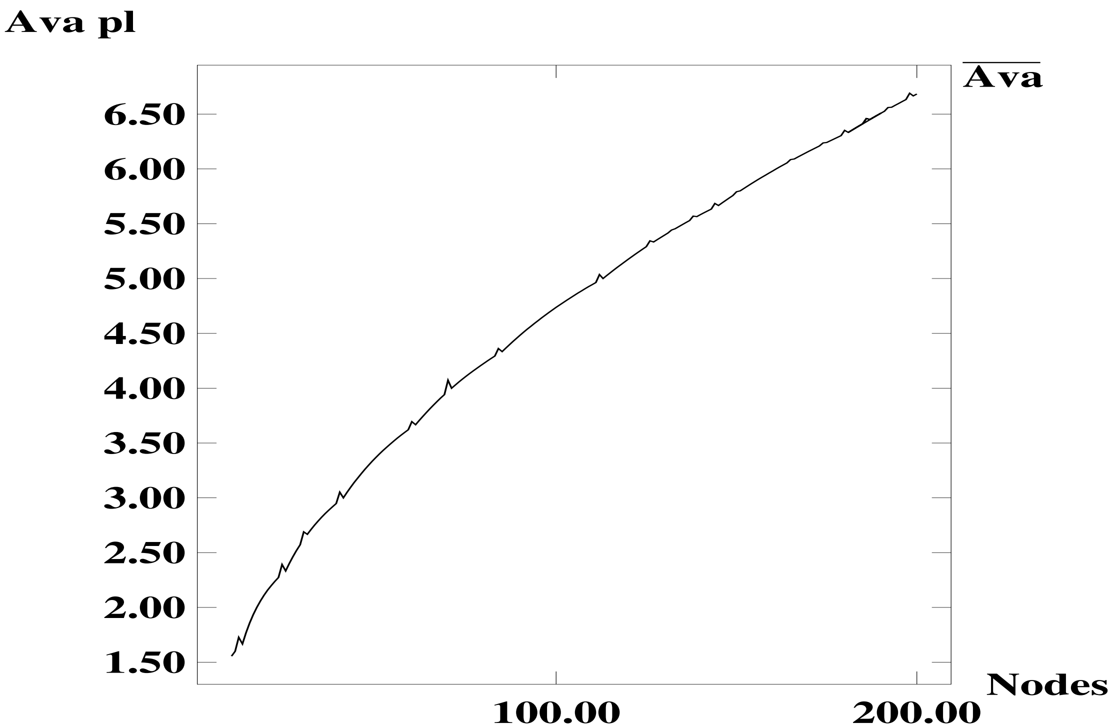


Figure 4.3: Average path length versus number of nodes in optimal loops

Chapter 5

MULTIPLE TOKEN MEDIA ACCESS IN MCDL

In this chapter we view MCDL as a union of several independent token rings with possible switching of data from one ring to another. We present delay-throughput performance through queueing models and compare them with the results of single ring [SS85] and multiple ring [BGY89]. We present in our approach, a multiple token media access control [CB98f], three switching mechanisms for packet transfer and in the process use the shortest path results of Chapter 3. The main features of our results are that the delay-throughput performance is better than single ring and close to multiple ring even at high loads. We also present advantages of MCDL over multiple rings such as coalescence of tokens, fewer number of interfaces and higher utilisation of bandwidth. Some preliminary studies on communication paradigms such as *broadcast, unicast and multicast* can be found in [BBC95, CB97].

5.1 Routing model in MCDL

For the multiple token media access control in MCDL we consider a directed version of MCDL. A directed MCDL of M nodes $0, 1, 2, \ldots (M-1)$, has only $[+1]$ and $[+h]$ links. The $[+1]$ links form a loop ring or $1-$ring and similarly the $[+h]$ links form g number of hop rings called $h-$rings, where $g = GCD(M, h)$. For simplicity we consider that $g = h$. With such a configuration, there are h number of hop rings and each hop ring has $\frac{M}{h}$ number of nodes, and one loop ring. Later in Section 6.3 we relax this condition on hop size and investigate its effect on the performance of MCDL. Figure 5.1 is an example of $M = 16$ and $h = 4$.

A *walk* on the $[+h]$ links is called *hop traversal* and the one on the $[+1]$ links is called *loop traversal.* A shortest path between any two nodes is a combination of both hop and loop traversals or only one of them (refer Section 2.2). For example in a 50 node directed network with 5 as hop size, the shortest path from node 4 to node 23 consists of three hop traversals and four loop traversals. One such path is 4-9-14-19-20-21-22-23. Similarly from node 12 to 22, the shortest path consists of only two hop traversals and the path is 12-17-22. Obviously the order in which the hop and loop traversals are traveled is immaterial as far as reaching

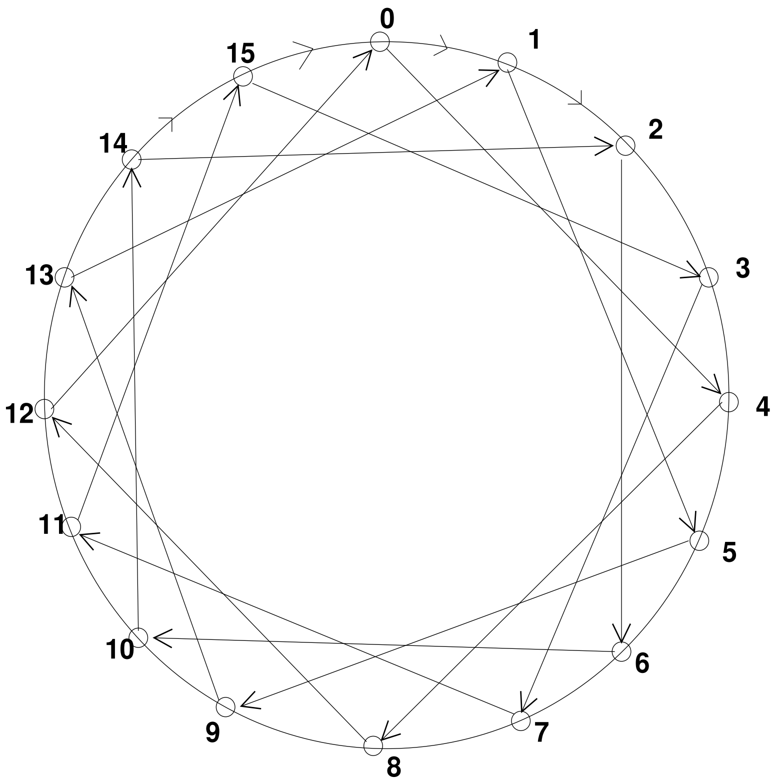


Figure 5.1: Directed MCDL with $M = 16$ and $h = 4$

the destination node. For example the three hop and four loop traversals to reach node 23 from 4, can be traveled as 4-5-10-11-16-17-22-23 or 4-5-6-11-16-17-18-23 *etc.* Algorithms to find shortest paths in MCDL networks have been developed in Chapter 3. We use shortest path routing of packets for the multiple token media access control.

5.2 Traffic models in MCDL

Every node in the directed MCDL is connected to a hop ring and the loop ring with respective ring interfaces. We classify the traffic that arrives at a node into three types: $h0-$traffic, $01-$traffic and $h1-$traffic depending on the destination addresses. The $h0-$traffic has only hop traversals, the $01-$traffic has only loop traversals and the $h1-$traffic has both on hop and loop ring traversals. For example, a packet whose source-destination addresses differ by multiples of hop size (in a shortest path), belongs to the $h0-$traffic. Similarly $01-$traffic contains those packets whose source-destination addresses differ by a value less than hop size. All other packets belong to the $h1-$traffic and have both the hop and loop traversals. We also refer to packets belonging to the $h1-$traffic as a $h1-$packet. Similarly we refer to $h0-$packets and $01-$packets.

We consider three types of routing the $h1-$traffic: *hop to loop routing (HTL)*, *loop to hop routing (LTH)* and *no-switching (NOS) routing*. In HTL, the $h1-$packet first completes hop traversal and later its loop traversals. In the LTH, the $h1-$packet is first completes loop traversals and later its hop traversals. In the case of NOS, the $h1-$packet is transferred on loop ring until its destination.

The structure of node buffers in HTL and LTH routing are shown in Figure 5.2 and 5.3 respectively. For storing the $h0$, 01 and $h1$ traffic types, we consider two types of buffers at each station: the $1-$buffer and the $h-$buffer. The $1-$buffer output is connected to the $1-$ring outgoing link at the node and the $h-$buffer to the outgoing link of the $h-$ring. The $01-$traffic that arrives from the host is stored in $1-$buffer as it needs only loop ring. Similarly, the $h0-$ traffic that arrives is stored in $h-$buffer as it needs only hop ring. Since the $h1-$traffic uses both hop and loop traversals, it can be stored either in the $1-$buffer or in the $h-$buffer, based on its routing.

In HTL routing (refer Figure 5.2), the $h1-$traffic that is generated by the host is stored in the $h-$buffer at the sender, since it is first transmitted on the hop ring for the complete required hop traversals. Let the node reached after completing these hop traversals be the *intermediate* node. At this intermediate node the $h1-$ traffic is *switched* to the local $1-$buffer as explained in the following. As a packet is received from the hop ring at the intermediate node, the hop token control shown in the Figure 5.2 identifies the destination address in it. If the destination address matches with the node address then it is the $h0-$packet and is transferred to the local host. Otherwise it is $h1-$packet, and is recognised as a *switching* packet. Such switching packets at the intermediate node form a *switch over* traffic. The switch over traffic at the intermediate node joins the $01-$traffic generated locally and both are together stored in its $1-$buffer. The switch over traffic is transferred from this $1-$buffer like the locally generated $01-$traffic on to the $1-$ring. We call this *hop to loop (HTL) switching.*

Alternatively in the *Loop to hop (LTH)* switching [CB98d] scheme (refer Figure 5.3), we store the $h1-$ traffic initially in the $1-$buffer at the sender and switch it to $h-$buffer at an appropriate *intermediate* node after completing all the required number of loop traversals. At this intermediate node the $h1-$traffic joins with the locally generated $h0-$traffic and completes the remaining number of hop traversals. Thus the $h1-$traffic switches from loop ring to hop ring in LTH routing.

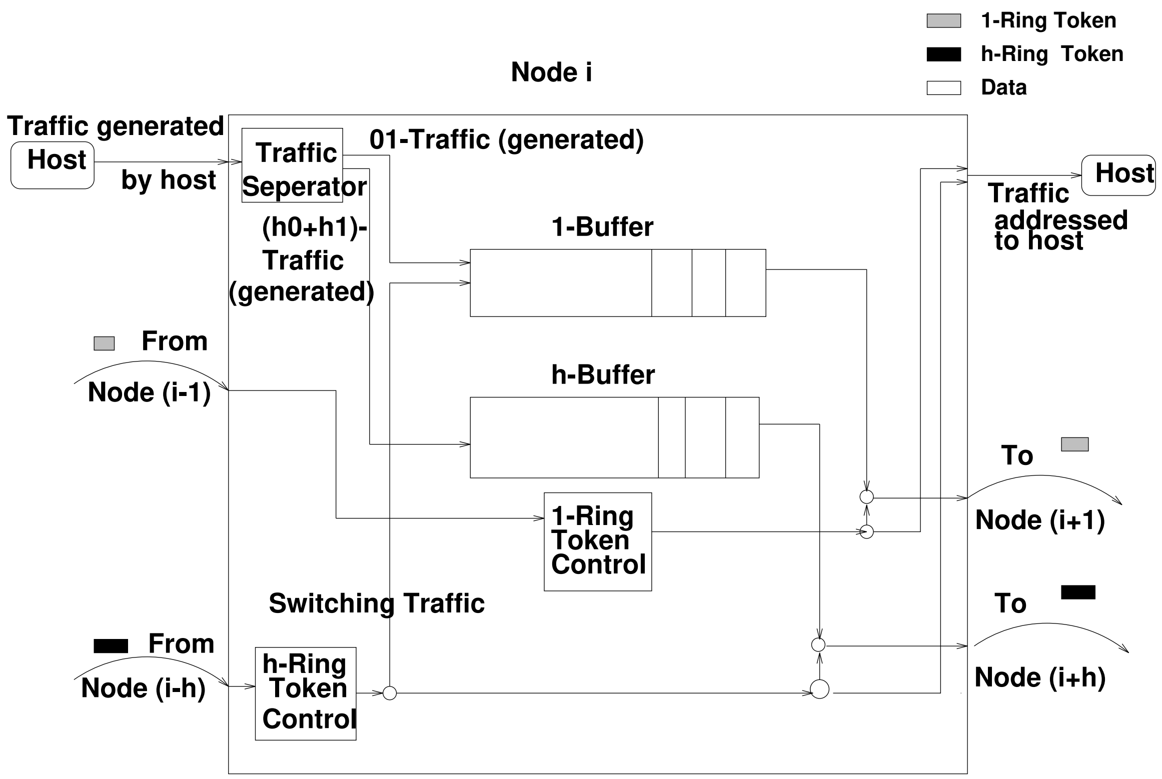


Figure 5.2: The internal block diagram of a node in HTL switching

In the NOS routing, the $h1-$traffic will be transferred up to the destination node on $1-$ring only, $i.e$, without switching at any intermediate node. So in this scheme the $h1-$traffic is stored in the $1-$buffer at the sender along with the $01-$traffic. Thus both the $01-$ and $h1-$traffics are transmitted on the loop ring. However, the $h0-$traffic is stored in the $h-$buffer and is transmitted on $h-$ring. This NOS model clearly distinguishes from the other two models in terms of the effect of intermediate node buffering and switching.

In all the three routing models we compute the shortest path according to the algorithms given in Chapter 3 and route them.

5.3 Media access protocol in MCDL

In Section 2.4.5 and 2.4.6, we reviewed token access and packet delay analysis in single ring (SRST) [SS85] and multiple rings (MRMT) systems [BGY89]. In our present work, we consider *multiple token media access* in MCDL. As mentioned earlier we view MCDL as $(h+1)$ independent rings. These are h number of hop rings and one loop ring. Each of these rings has a token for media access control. Each of these hop and loop ring tokens grant medium access

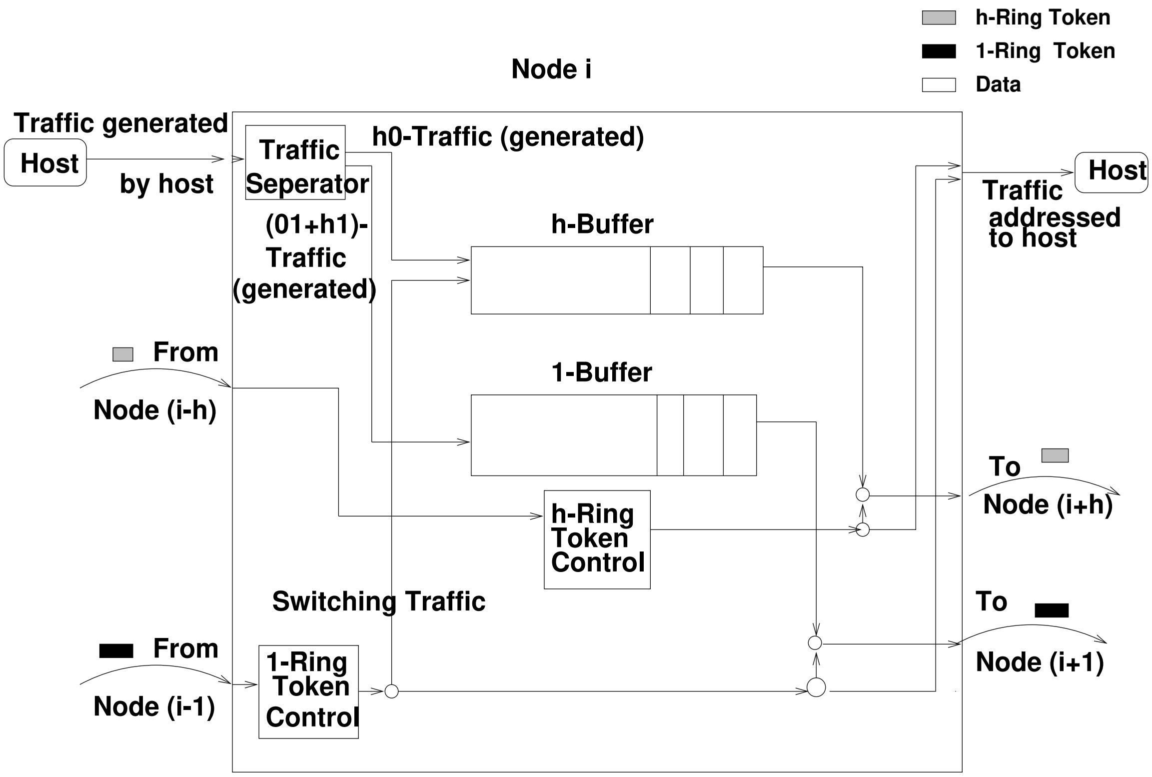


Figure 5.3: The internal block diagram of a node in LTH switching

rights in the same way as SRST model. Thus we implement the same media access protocol in MCDL to compare the performance of packet delays in SRST, MRMT and MCDL models.

The media access control in MCDL and MRMT differ in a subtle way. In MRMT when a node receives more than one token simultaneously or if it receives a token while already in possession of one, then the node simply passes on the remaining tokens other than the one it is currently using for transmitting a packet to the next node on the ring. It is done so in MRMT to avoid any buffer conflicts in the node. Whereas in MCDL such buffer conflicts do not arise as the traffics have been classified at the source and accordingly they are stored in separate buffers. In view of buffer conflicts in MRMT, only one among the B tokens grant medium access right to a node for one of the B channels. Whereas in MCDL in view of no buffer conflicts, at most two of the $(h+1)$ tokens can at any time grant medium access right to a node for both the (hop and loop) channels. Thus we see no token is unutilised. In MRMT and SRST a packet suffers waiting only at the sender. But as per the HTL and LTH routing in MCDL, some packets suffer waiting time at two nodes before reaching the destination. However packets are routed along the shortest path in MCDL. Thus we wish to investigate the additional queueing stage in MCDL against its shortest path routing. We also investigate the simultaneous use of two channels in MCDL against using of only one channel in MRMT.

Another difference between MRMT and MCDL is that in MRMT the B tokens are likely to move together on the rings [VR94, Kam94] resulting in coalescence of tokens. In such situation the MRMT is functioning like SRST. But in MCDL the tokens will never coalesce.

5.4 Modeling switch over traffic

As given in the analysis of SRST and MRMT, the hop ring token is the server for the $h-$buffer and loop ring token is the server for the $1-$buffer at a node. The server (token) visits each station, picks a packet for transmission if it has one and goes to the the next and so on, rotating round the corresponding ring. In each rotation it grants every node to transmit a packet (*non-exhaustive* or *1-limited* service). So the rotation time of a token is the *service time* with respect to any node. The successive service times vary randomly since the number of nodes that use the server in a rotation vary. So we say that the service time of any server (hop or loop token) is *generally distributed*. In the following we consider the switch over traffic in both the HTL and LTH cases at the intermediate node.

In the case of HTL switching (modelled in Figure 5.4), the traffic that enters into $1-$buffer of intermediate node is denoted as the X^s-traffic and its moments are not obtainable straight away. The $01-$traffic is the locally host generated traffic and is Poisson. Thus the problem lies in modeling the χ_s-traffic distribution which is the combination of X^s- and $01-$ traffics. Since the loop token service time is *generally distributed*, the $1-$buffer is an $A/G/1$ queue where A is some distribution of χ_s-traffic. The modeling and its analysis of χ_s-traffic is given in Section 5.4.1.

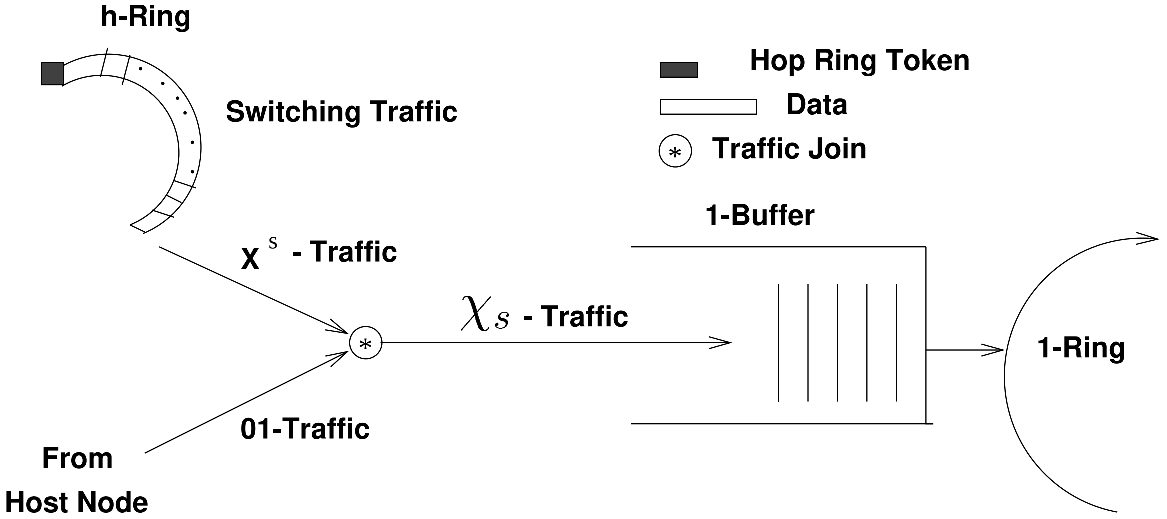


Figure 5.4: The switch over traffic model at 1-ring queue in HTL

With similar argument, the $h-$buffer in LTH switching is also an $A/G/1$ queue (refer Figure 5.5) where A is some distribution of χ_s-traffic which is the combination of switching traffic (X^s-traffic) from loop ring and $h0-$traffic from the local host. The modeling and analysis of switch over traffic in LTH switching is given on Section 5.4.2. The traffic destined to the intermediate node (in both the switching models) is not shown in the Figures 5.4 and 5.5 as it does not add to the switching traffics. We analyse both the switch overs here.

The first step in our analysis is to find the joint distribution of χ_s-traffic and its moments. These results are further useful in finding the waiting time of the switched packets with the help of the queueing formulae. In the following, first we introduce the notations used in our analysis.

M Number of nodes.

r Node-to-node propagation time including node processing time of the token.

h Hop size. We assume $g = GCD(M, h) = h$.

x Packet transmission time in units of bit times. (So x is also packet length in bits). Unless otherwise mentioned, we assume fixed length packets.

λ Average traffic arrival rate at a node from host. We assume Poisson arrival.

R_h Round trip propagation time of hop ring. $R_h = \frac{M}{h} \cdot r$.

R_1 Round trip propagation time of loop ring. $R_1 = M \cdot r$.

λ_{h0} Average $h0-$traffic generated at a node by the host.

λ_{01} Average $01-$traffic generated at a node by the host.

λ_{h1} Average $h1-$traffic generated at a node by the host.

When $h = g$, a hop ring has $\frac{M}{h}$ number of nodes. The shortest path from a node of the hop ring to the other $\frac{M}{h} - 1$ nodes of this ring contains only $[+h]$ links. Similarly the shortest path from a node to the following $(h-1)$ nodes on the loop ring contains only $[+1]$ links. All the remaining nodes given by $(M-1)-(\frac{M}{h}-1)-(h-1) = (\frac{M}{h}-1)(h-1)$ have both the $[+h]$ and $[+1]$ links. From this and the assumption of uniform traffic *i.e*, all destinations are equally likely for a generated packet, we have

$$\lambda_{h0} = \frac{\frac{M}{h} - 1}{M - 1} \cdot \lambda \tag{5.1}$$

$$\lambda_{01} = \frac{h - 1}{M - 1} \cdot \lambda \tag{5.2}$$

$$\lambda_{h1} = \frac{(\frac{M}{h} - 1)(h - 1)}{M - 1} \cdot \lambda \tag{5.3}$$

Since λ_{h0}, λ_{01}, and λ_{h1}, are fixed fractions of λ, they are all Poisson. We also note that

$$\lambda = \lambda_{h0} + \lambda_{01} + \lambda_{h1}.$$

λ_h Average traffic that enters the hop buffer in HTL routing .

λ_1 Average traffic that enters the loop buffer in LTH routing.

In case of HTL switching, $h0-$ and $h1-$traffics enter the the $h-$buffer. Hence $\lambda_h = \lambda_{h0} + \lambda_{h1}$. Similarly in LTH switching, $01-$ and $h1-$ traffics enter the $1-$buffer. Hence $\lambda_1 = \lambda_{01} + \lambda_{h1}$. So we have from (5.1), (5.2) and (5.3)

$$\lambda_h = \lambda_{h0} + \lambda_{h1} = \frac{h(\frac{M}{h} - 1)}{M - 1} \cdot \lambda \tag{5.4}$$

$$\lambda_1 = \lambda_{01} + \lambda_{h1} = \frac{\frac{M}{h}(h - 1)}{M - 1} \cdot \lambda \tag{5.5}$$

N_h The number of active stations on a hop ring .

N_1 The number of active stations on the loop ring.

A station on a hop (or loop) ring is active if at any time it has at least one packet waiting in its $h-$ (or $1-$) buffers. It is assumed that the number of active stations is a *binomially distributed* random number [SS85]. Mean of N_h and N_1 are denoted by $\overline{N_h}$ and $\overline{N_1}$ respectively.

T_h Cycle time of hop ring token.

T_1 Cycle time of loop ring token.

Cycel time is the time between a token leaving a station, making one complete round on the corresponding ring and reaching the same station again. In each rotation of the token, one packet from the corresponding buffers is transmitted from an active station. So,

$$T_h = N_h \cdot x + \frac{M}{h} \cdot r \tag{5.6}$$

$$T_1 = N_1 \cdot x + M \cdot r \tag{5.7}$$

The maen of T_h and T_1 are denoted by $\overline{T_h}$ and $\overline{T_1}$ respectively.

ρ_h Probability that a node on a hop ring is active.

ρ_1 Probability that a node on the loop ring is active.

In HTL switching mode from (C.5),

$$\rho_h = \lambda_h \cdot \overline{T_h} \tag{5.8}$$

$$\rho_1 = \overline{\chi_s} \cdot \overline{T_1} \tag{5.9}$$

In LTH switching mode from (C.5),

$$\rho_h = \overline{\chi_s} \cdot \overline{T_h} \tag{5.10}$$

$$\rho_1 = \lambda_1 \cdot \overline{T_1} \tag{5.11}$$

5.4.1 Mean and variance of switching traffic in HTL routing

As explained in Section 5.4, the switching traffic χ_s in HTL switching is the combination of X^s- and $01-$traffics. Our interest lies in finding its joint distribution. For this purpose let us consider a hop ring. Since N_h is binomially distributed, from Table C.1 we get,

$$\overline{N_h} = \frac{M}{h} \cdot \rho_h$$

We get from (5.6) and (5.8)

$$\begin{aligned}\overline{T_h} &= \overline{N_h} \cdot x + \frac{M}{h} \cdot r \\ &= \frac{\frac{M}{h} \cdot r}{1 - \frac{M}{h} \cdot \lambda_h \cdot x} \qquad (5.12)\end{aligned}$$

Average number of $h-$ring token rotations per unit time, n is given by,

$$n = \frac{1 - \frac{M}{h} \cdot \lambda_h \cdot x}{\frac{M}{h} \cdot r} \qquad (5.13)$$

Let P_i be the number of packets that are carried by the $h-$ring token in its i^{th} $(1 \leq i \leq n)$ rotation.

$$Pr[P_i = k] = ({}^{\frac{M}{h}}C_k)\ \rho_h^k \cdot (1 - \rho_h)^{(\frac{M}{h} - k)},$$

whose $p.g.f$ from Table C.1 is

$$P_i(s) = (\rho_h \cdot s + 1 - \rho_h)^{\frac{M}{h}} \qquad (5.14)$$

Let $P =$ total packets carried by the $h-$ring token in a unit time of n rotations

$$P = P_1 + P_2 + \ldots + P_n,$$

Assuming that P_i's are independent of each other and also of n, the $p.g.f$ of P using Theorem-C.1, (5.13) and (5.14) is

$$\begin{aligned}P(s) &= [(\rho_h \cdot s + 1 - \rho_h)^{\frac{M}{h}}]^{\frac{1 - \frac{M}{h}\lambda_h x}{\frac{M}{h} r}} \\ &= (\rho_h \cdot s + 1 - \rho_h)^{\frac{1 - \frac{M}{h}\lambda_h x}{r}} \qquad (5.15)\end{aligned}$$

The mean and variance of P from (C.1) and (C.3) are,

$$\begin{aligned}\overline{P} &= [\frac{dP(s)}{ds}]_{(s=1)} \\ &= \frac{M}{h} \cdot \lambda_h \qquad (5.16) \\ \sigma_P^2 &= [\frac{d^2P(s)}{ds^2}]_{(s=1)} + [\frac{dP(s)}{ds}]_{(s=1)} - ([\frac{dP(s)}{ds}]_{(s=1)})^2 \\ &= (\frac{M}{h} \cdot \lambda_h)^2 [\frac{1}{\frac{M}{h}\lambda_h} - \frac{r}{1 - \frac{M}{h}\lambda_h x}] \qquad (5.17)\end{aligned}$$

The σ_P^2 in (5.17) can be approximated by noting that the interarrival time of $h-$buffer packets $(\frac{1}{\lambda_h})$ must be more than the maximum hop token rotation time. Otherwise, the $h-$buffer will overflow and be unstable. The T_h is maximum when N_h in (5.6) is $\frac{M}{h}$. Hence in the stable range, we get

$$\frac{1}{\lambda_h} > \frac{M}{h}x + \frac{M}{h}r$$

$$\begin{aligned} 1-\frac{M}{h}\lambda_h x &> \frac{M}{h}\lambda_h r \\ \frac{1}{\frac{M}{h}\lambda_h} &> \frac{r}{1-\frac{M}{h}\lambda_h x} \end{aligned} \tag{5.18}$$

Using (5.18) in (5.17), we can approximate

$$\sigma_P^2 \approx \overline{P} \tag{5.19}$$

Since mean and variance of P are approximately equal, P may be approximated by a Poisson process.

Assuming that the carried packets P by the $h-$ring token are delivered uniformly at every node on the $h-$ ring, then those delivered at a hop node are $P_d = \frac{P}{\frac{M}{h}}$. The delivered packets consists of those that are addressed to this intermediate node and those that are addressed to the next $(h-1)$ nodes on the loop ring (for switch over). Assuming that the delivered packets P_d are uniformly distributed among these h nodes, we can say that $\frac{1}{h}$ are addressed to the intermediate node and $\frac{h-1}{h}$ are for switch over. Thus

$$\begin{aligned} X^s &= \frac{h-1}{h}\cdot P_d \\ &= \frac{h-1}{h}\cdot\frac{1}{\frac{M}{h}}\cdot P \end{aligned} \tag{5.20}$$

Since X^s is a scalar multiple of P that is approximated as Poisson, we can say that X^s is also approximately Poisson. Since $01-$traffic is Poison and X^s is approximated as Poisson, then χ_s which is the combination of these two is also approximately Poisson. The average rate of χ_s-traffic is

$$\begin{aligned} \overline{\chi_s} &= \overline{X^s} + \lambda_{01} \\ &= \frac{h-1}{h}\cdot\frac{1}{\frac{M}{h}}\cdot\frac{M}{h}\cdot\lambda_h + \lambda_{01} \\ &= \frac{(h-1)(\frac{M}{h}-1)}{M-1}\cdot\lambda + \lambda_{01} \\ &= \lambda_{h1} + \lambda_{01} \end{aligned} \tag{5.21}$$

Thus the input to the loop buffer in HTL switching is modeled as Poisson process with an average arrival rate given by (5.21).

5.4.2 Mean and variance of switching traffic in LTH routing

Here we consider the case that the switch over traffic travels first on $1-$ring and later switch to $h-$ring at an intermediate node (refer Figure 5.5). As done in the analysis of HTL switching in Section 5.4.1, we find $\overline{\chi_s}$ the mean number of packets that switch to $h-$buffer and its

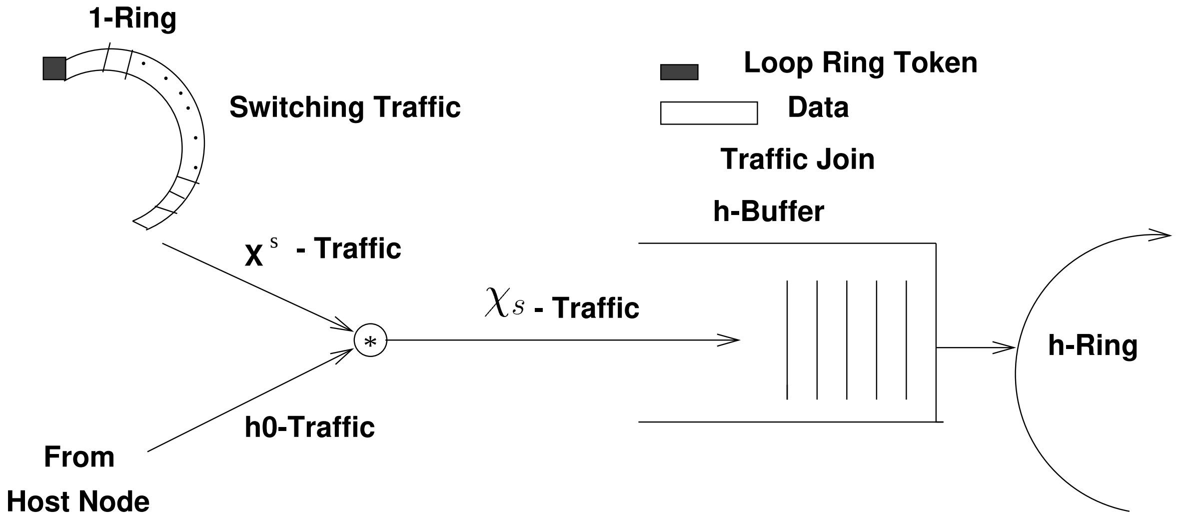


Figure 5.5: The switch over traffic model at h-ring queue in LTH

distribution. Similar to equation(5.12), the mean rotation time of loop token is:

$$\overline{T_1} = \frac{M \cdot r}{1 - M \cdot \lambda_1 \cdot x} \tag{5.22}$$

Similar to equation (5.15), the mean number of rotations of loop token per unit time is:

$$n = \frac{1 - M \cdot \lambda_1 \cdot x}{M \cdot r} \tag{5.23}$$

Let P_i be the number of packets that are carried by the $1-$ring token in its i^{th} rotation. Then the probability generating function $p.g.f$ of P is

$$P(s) = (\rho_1 \cdot s + 1 - \rho_1)^M$$

The mean and variance of P are obtained from (C.1) and (C.3)

$$\overline{P} = M \cdot \lambda_1 \tag{5.24}$$

$$\sigma_P^2 = (M \cdot \lambda_1)^2 [\frac{1}{M\lambda_1} - \frac{r}{1 - M\lambda_1 x}] \tag{5.25}$$

Similar to the assumption made for (5.18), we can prove that mean and variance of P are equal, resulting

$$\overline{P} \approx \sigma_P^2 \tag{5.26}$$

Hence P is approximately Poisson. Similar to (5.20), the packets switched to $h-$buffer χ_s can also be proved as Poisson. Finally the mean of switch over traffic into the $h-$buffer is,

$$\overline{\chi_s} = \lambda_{h0} + \lambda_{h1} \tag{5.27}$$

Thus the input to the $h-$buffer in LTH switching is also be modeled as approximately Poisson process.

Chapter 6

DELAYS IN MCDL

The results that the switch over traffic may be approximated by a Poisson distribution (from Chapter 5) help us now to model the $h1-$traffic. The $h1-$traffic passes through two queues: at the sender and at the intermediate node. However, the $h0-$ and $01-$traffics travel through only one queue at the sender. The queue at the sender is an $M/G/1$ queue [SS85] for the $h1-$traffic. The queue at the intermediate node for the switch over traffic is approximated by an $M/G/1$ queue (refer Section 5.4). We obtain the *average delay* of a packet [CB99a] by knowing the weights of $h0$, 01 and $h1$ traffics and the $M/G/1$ waiting times from [Kle75] given in (C.6).

6.1 Modeling of delay

We define the following from (5.4) and (5.5):
Weight (or fraction) of h-ring traffic generated by the host

$$\begin{aligned} w_h &= \frac{\lambda_h}{\lambda} \\ &= \frac{h(\frac{M}{h}-1)}{M-1} \end{aligned} \tag{6.1}$$

Weight (or fraction) of 1-ring traffic generated by the host

$$\begin{aligned} w_1 &= \frac{\lambda_1}{\lambda} \\ &= \frac{\frac{M}{h}(h-1)}{M-1} \end{aligned} \tag{6.2}$$

The hop ring has $\frac{M}{h}$ number of nodes. A packet on the average finds its destination $\frac{1}{2} \cdot \frac{M}{h}$ stations away. Since the node to node to propagation is assumed to be r, the average propagation time of a packet is $\frac{M}{h} \cdot \frac{1}{2} \cdot r$.

The average propagation time of a packet on the $1-$ring is as follows. The average path length that a packet travels on the loop ring is $\frac{h}{2}$ as any packet has a maximum number loop traversals equal to $(h-1)$.

W_h	Waiting time of a packet in hop buffer.
W_1	Waiting time of a packet in loop buffer.
D_h	Delay of a packet transmitted on to the $h-$ring.
D_1	Delay of a packet transmitted on to the $1-$ring.
D_a	Weighted average Delay of a packet. The weighted average delay is defined:

$$D_a = w_h \cdot D_h + w_1 \cdot D_1 \tag{6.3}$$

The waiting time of a packet in hop buffer is defined as the time between when the packet arrives hop buffer for transmission and the time it starts being transmitted.

The waiting time of a packet in loop buffer is defined as the time between when the packet arrives loop buffer for transmission and the time it starts being transmitted.

The $h-$ring delay is defined as the sum of waiting time in the $h-$buffer (W_h), packet transmission time (x) and average propagation time for a packet in completing its required hop traversals $(\frac{M}{h} \cdot r \cdot \frac{1}{2})$.

The $1-$ring delay is defined as the sum of waiting time in the $1-$buffer (W_1), packet transmission time (x) and average propagation time for a packet in completing its required loop traversals $(\frac{h}{2} \cdot r)$.

We present in the following the weighted average delays (or simply average delays) analytically from the media access and switching models presented in Chapter 5. We also use queueing models explained in Appendix-C.

6.1.1 Average delay in HTL switching

The $h-$buffer is an $M/G/1$ queue. We proved that the switch over traffic is approximately Poisson in Section 5.4.1. Hence the $1-$buffer input is also Poisson. So the $1-$buffer is also an $M/G/1$ queue. So the complete model of the $h1-$traffic is a tandem of two $M/G/1$ queues. The $h0-$traffic passes through the $M/G/1$ queue of $h-$buffer and the $01-$traffic passes through the $M/G/1$ queue of $1-$buffer. In order to find the *weighted average delay* of packets in the HTL switching, we refer to equation (6.3). The D_h and D_1 in this are obtained below.

The delay of packets in the hop buffer D_h is obtained from the $M/G/1$ model of the $h-$buffer. We saw in equation 2.12 the waiting time and in equation (2.13) the total packet delay for an M node SRST whose node buffers are modeled as $M/G/1$ queue. By substituting $\frac{M}{h}$ in place of M in these equations, we get the waiting time and delay in the hop buffer at a node in MCDL. Thus:

$$D_h = \frac{M/h}{2}[\frac{r}{1-(M/h)\lambda_h \cdot x - (M/h)\lambda_h \cdot r} + x^2\lambda_h] + x + \frac{M}{h} \cdot r \cdot \frac{1}{2} \tag{6.4}$$

The λ_h, x, r, are the usual definitions. The first term accounts for the waiting time in the $h-$buffer obtained from equation (2.12) and the second term is due to transmission time for the packet of length x. The last term accounts for the average propagation time of the packet on the hop ring.

Now we derive W_1, the waiting time of packets in $1-$buffer which has χ_s-traffic arrival and was shown to be approximately Poisson, refer equation (5.21. Thus the $1-$buffer is also an $M/G/1$ queue. The service time of a packet in $1-$buffer (T_1) and the traffic intensity of a $1-$buffer (ρ_1) are from (5.7) and (5.9). They are:

$$\begin{aligned} T_1 &= N_1 x + Mr \\ \rho_1 &= \overline{\chi_s} \cdot \overline{T_1} \end{aligned}$$

Since N_1 is binomially distributed, T_1 is a generally distributed random variable. Thus the $1-$buffer is an $M/G/1$ queue. For the $1-$buffer delay, similar to the derivation of (2.12), we obtain

$$\begin{aligned} N_1(s) &= (\rho_1 s + 1 - \rho_1)^M \\ T_1(s) &= s^{Mr}(\rho_1 s + 1 - \rho_1)^M \\ \overline{N_1} &= M \cdot \rho_1 \\ \sigma^2_{N_1} &= M\rho_1(1-\rho_1) \\ \overline{T_1} &= M\rho_1 x + Mr \\ \sigma^2_{T_1} &= M\rho_1(1-\rho_1)x^2 \end{aligned}$$

Using ρ_1 in $\overline{T_1}$ and simplifying, we get

$$\overline{T_1} = \frac{Mr}{1 - M\overline{\chi_s}x} \tag{6.5}$$

Now using (C.6), we get the delay of packets in $1-$buffer as

$$D_1 = \frac{M}{2}[\frac{r}{1 - M\overline{\chi_s} \cdot x - M\overline{\chi_s} \cdot r} + x^2\overline{\chi_s}\ ] + x + \frac{h}{2} \cdot r \tag{6.6}$$

In the above, similar to the delay expression in (6.4) the first term is the waiting time, second term is the transmission time and the last is the average propagation time on the loop ring. The $\overline{\chi_s}$ is given in (5.21). Using (6.1), (6.2),(6.4) and (6.6) in (6.3), the weighted average delay D_a of packets is obtained. We have obtained weighted average delay for different values

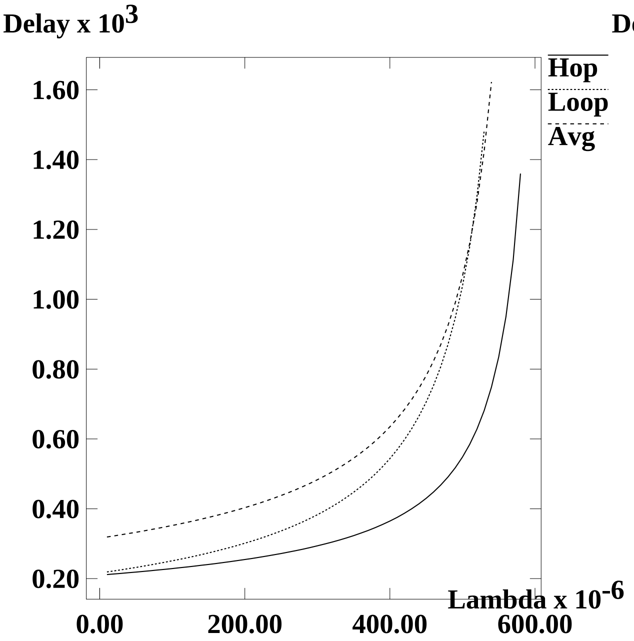


Figure 6.1: Hop, loop and avg delays in HTL h=2, M=50

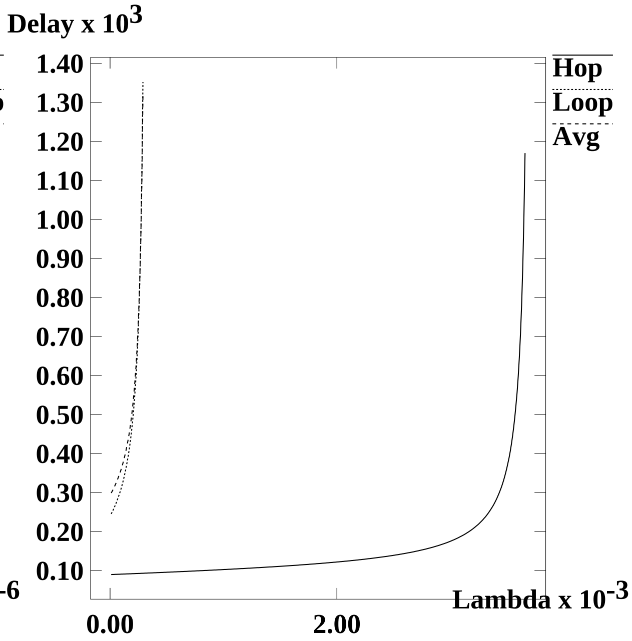


Figure 6.3: Hop, loop and avg delays in HTL h=10, M=50

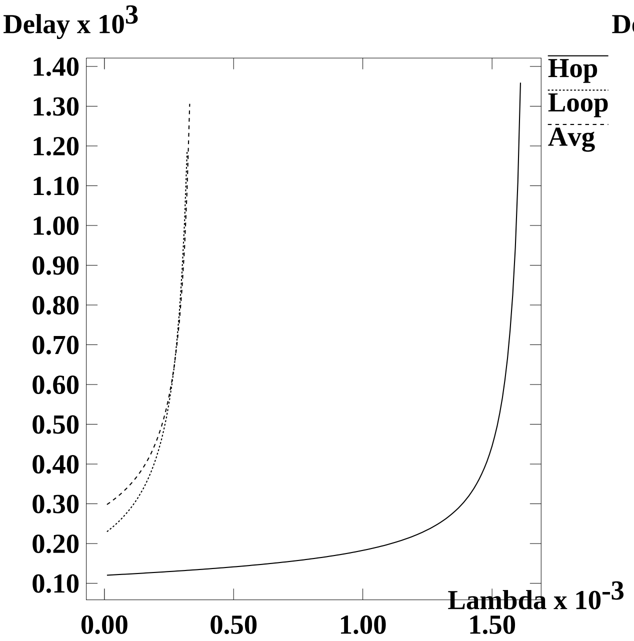


Figure 6.2: Hop, loop and avg delays in HTL h=5, M=50

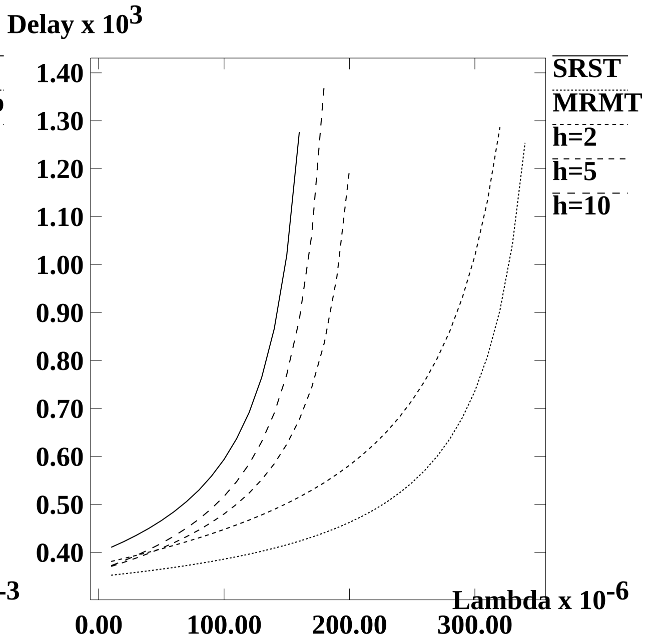


Figure 6.4: Avg delay in HTL (h=2, 5 and 10), SRST, MCDL

of λ at different hop sizes. Then we compared these values with delays of MRMT model given in (2.14) and SRST model (2.13). For comparing the delays of MCDL with these two networks, the following network characteristics are used. Number of nodes=50, hop sizes=2,5 and 10, packet size=100 bits (fixed) and node-to-node propagation time=6 bits. All time calculation are done in bit time at 10^5 bits/sec. While these assumptions may be unrealistic for a real network, they allow for comparative studies to be made. In case of MRMT we choose to ompare with $B = 2$ since the degree of MRMT and that of MCDL are same for this value of B. The delays are given in Figures 6.1 through 6.4.

6.1.2 Average delay in LTH switching

The delay D_1 can be obtained from (2.12) and is given by

$$D_1 = \frac{M}{2}[\frac{r}{1 - M\lambda_1 \cdot x - M\lambda_1 \cdot r} + x^2\lambda_1] + x + \frac{h}{2} \cdot r \tag{6.7}$$

Similar to the HTL switching, the T_h and ρ_h are

$$\begin{aligned} T_h &= N_h x + \frac{M}{h} r \\ \rho_h &= \overline{\chi_s} \cdot \overline{T_h} \end{aligned}$$

The N_h is binomially distributed, T_h is a generally distributed random process and hence the $h-$buffer is an $M/G/1$ queue. Thus in this switching also the $h1-$traffic passes through a tandem of $M/G/1$ queues: the $1-$buffer (M/G/1 queue) to $h-$buffer (M/G/1 queue). The delay in $1-$buffer was given in (6.7). Similar to (6.6), we obtain the following to find D_h:

$$\begin{aligned} N_h(s) &= (\rho_h s + 1 - \rho_h)^{\frac{M}{h}} \\ T_h(s) &= s^{\frac{M}{h}r}(\rho_h s + 1 - \rho_h)^{\frac{M}{h}} \\ \overline{N_h} &= \frac{M}{h} \cdot \rho_h \\ \sigma^2_{N_h} &= \frac{M}{h}\rho_h(1 - \rho_h) \\ \overline{T_h} &= \frac{M}{h}\rho_h x + \frac{M}{h} r \\ \sigma^2_{T_h} &= \frac{M}{h}\rho_h(1 - \rho_h)x^2 \end{aligned}$$

Using ρ_h in $\overline{T_h}$ and simplifying we get

$$\overline{T_h} = \frac{\frac{M}{h}r}{1 - \frac{M}{h}\overline{\chi_s}x} \tag{6.8}$$

Now using (C.6), we get the D_h the delay of packets in $h-$buffer, which has χ_s-traffic arrival as shown in Figure 5.5.

$$D_h = \frac{M/h}{2}[\frac{r}{1 - (M/h)\overline{\chi_s} \cdot x - (M/h)\overline{\chi_s} \cdot r} + x^2\overline{\chi_s}\,] + x + \frac{M}{h} \cdot r \cdot \frac{1}{2} \tag{6.9}$$

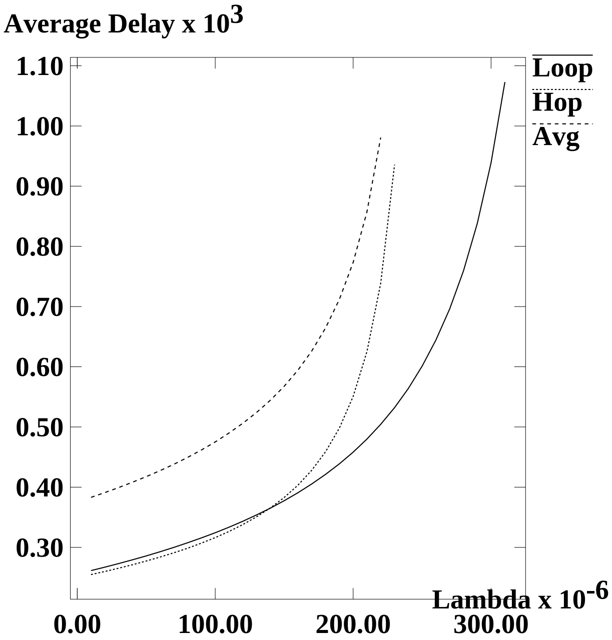


Figure 6.5: Hop, loop and avg delays in LTH h=2, M=50

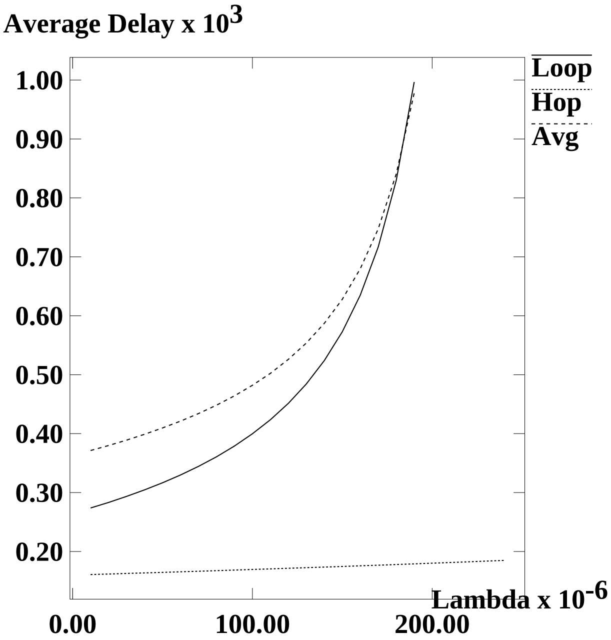


Figure 6.7: Hop, loop and avg delays in LTH h=10, M=50

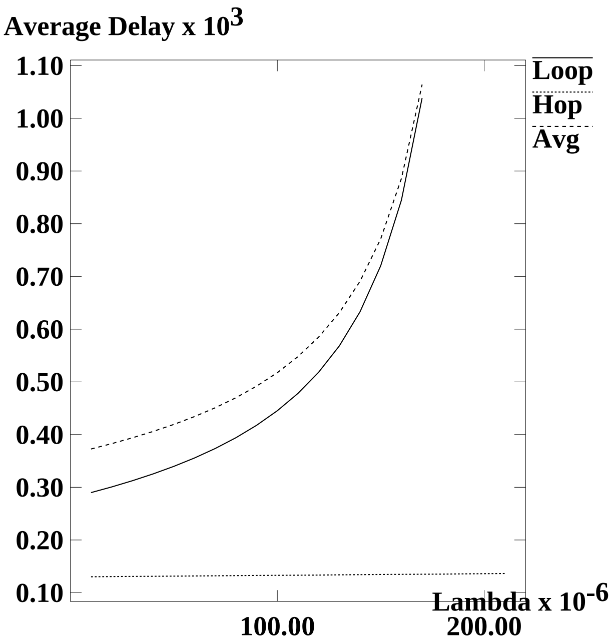


Figure 6.6: Hop, loop and avg delays in LTH h=5, M=50

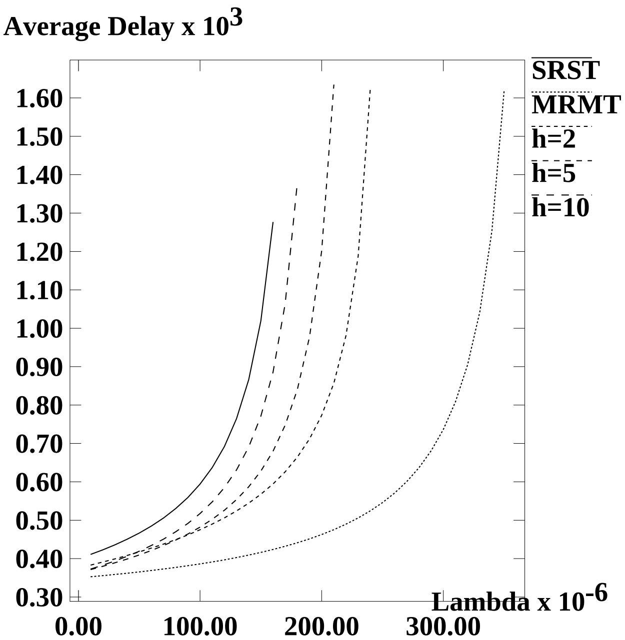


Figure 6.8: Avg delay in LTH (h=2, 5 and 10), SRST, MCDL

The above contains waiting time in $h-$buffer, transmission time and average propagation time on $h-$ring. Using the weights w_h, w_1 from (6.1) and (6.2), delays D_1, D_h from (6.7) and (6.9), and using the formula for D_a given by equation (6.3), the weighted average delay is computed. We took $M = 50$, $x = 100$ and $r = 6$. All the values of the times are expressed in bit time. The plots are given in Figures 6.5 through 6.8 comparing with SRST and MRMT with $B = 2$.

6.2 Throughput studies

We analyse the throughput to determine the total message traffic that a node can handle before saturation occurs to it. When buffer utilisation (*i.e*, ρ, the traffic intensity at a queue) reaches unity, the node is saturated. The total message arrival rate at which this occurs is the maximum traffic that a node can handle and is defined to be the throughput of that node. We compute the maximum traffic at a node by relating each buffer (hop and loop) utilisation (ρ_h and ρ_1 correspondingly) to the maximum arrival rate of that buffer and setting the maximum utilisation to one. In the following we derive the maximum traffic arrival rate at a node in both the HTL and LTH switchings.

6.2.1 Hop to loop switching performance

We characterized the HTL switching with the following parameters in Chapter 5:

$$\begin{aligned}
\lambda_h &= \frac{(\frac{M}{h}-1)\cdot h}{M-1}\cdot\lambda \\
\overline{\chi_s} &= \frac{\frac{M}{h}(h-1)}{M-1}\cdot\lambda \\
\overline{T_h} &= \frac{\frac{M}{h}r}{1-\frac{M}{h}\cdot\lambda_h\cdot x} \\
\overline{T_1} &= \frac{Mr}{1-M\cdot\overline{\chi_s}\cdot x}
\end{aligned}$$

We compute the maximum arrival rates for both the hop and loop buffers as follows. The maximum arrival rate of traffic at the hop buffer is obtained when its utilisation, (the hop buffer traffic intensity ρ_h) is one. Similarly for the loop buffer. So for the hop buffer we equate $\lambda_h\cdot\overline{T_h}$ to 1 and similarly for the loop buffer we equate $\overline{\chi_s}\cdot\overline{T_1}$ to 1. By simplifying these equations, we get λ_{uh} the maximum arrival rate for which the $h-$buffer is saturated and similarly λ_{u1} the maximum arrival rate at which the $1-$buffer is saturated. They are:

$$\begin{aligned}
\lambda_{u1} &= \frac{h}{h-1}\cdot\frac{M-1}{M}\cdot\frac{1}{M}\cdot\frac{1}{x+r} \\
\lambda_{uh} &= \frac{h}{M-h}\cdot\frac{M-1}{M}\cdot\frac{1}{x+r}
\end{aligned}$$

Table 6.1: Maximum traffic arrival rates for hop and loop buffers in HTL switching

Hop size	λ_h	$\overline{\chi_s}$	λ_{uh} $(pkts/sec)$	λ_{u1} $(pkts/sec)$
2	.9795 λ	.5102 λ	38.52	36.98
5	.9183 λ	.8163 λ	102.72	23.11
10	.8163 λ	.9183 λ	231.12	20.54
25	.5102 λ	.9795 λ	924.48	19.26

$$= \quad M \cdot \frac{h-1}{M-h} \cdot \lambda_{u1}$$

We tabulate in Table 6.1 the $h,\ \lambda_h,\ \overline{\chi_s},\ \lambda_{uh}$ and λ_{u1} for $M = 50,\ x = 100,\ r = 6$ at 10^5bps. The multiplicative factor for λ_{u1} in the expression for λ_{uh} is always greater than one for any value of hop size. Hence the loop becomes saturated at much lower value of λ than the hop buffer. Thus the loop ring load limits the overall network throughput. More over λ_{u1} decreases with increase in hop size. This is also in accordance with the fact that as hop size increases the number nodes on hop ring decreases, corresponding nodes on loop ring increase and thus loop traffic increases. So the loop token rotates slower than the hop token resulting in the loop buffer service becoming slower and reaching the saturated level at much lower $lambda'$s than hop buffer. These findings are observable from the Figure 6.9 also. The throughput at a

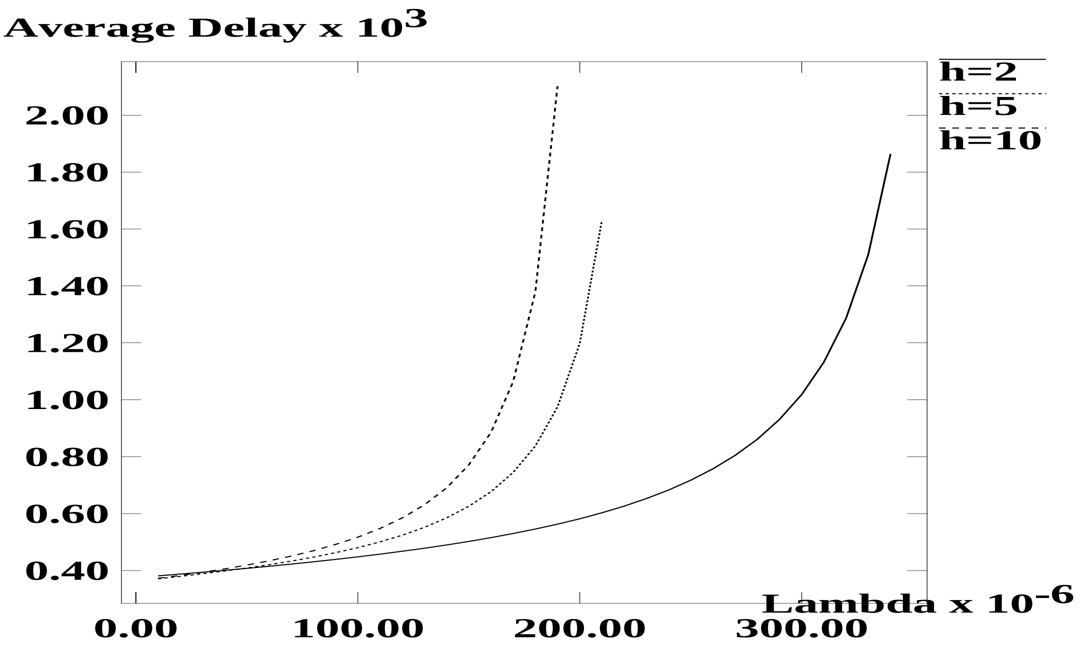


Figure 6.9: Delays in HTL showing decrease in throughput for diferent hop sizes (Table 6.1)

node is from column 4 of the Table 6.1. Similar values for SRST can be obtained by equating $\rho = 1$ where ρ is given in Section 2.12. By solving, we obtain the maximum arrival rate at a node buffer in SRST is 18.86 pkts/sec, which is the throughput at a node. Thus MCDL with hop connections increases node throughput than SRST. This increase is more when hop size is small.

6.2.2 Loop to hop switching performance

As in the case of HTL switching, the maximum arrival rates at hop and loop buffers are studied here. The maximum arrival rate for loop buffer is obtained when $\lambda_1 \cdot \overline{T_1} = 1$. Similarly the maximum arrival rate for the hop buffer is obtained when $\overline{\chi_s} \cdot \overline{T_h} = 1$. Simplifying we get,

$$\begin{aligned}
\lambda_{u1} &= \frac{h}{h-1} \cdot \frac{M-1}{M} \cdot \frac{1}{M \cdot (x+r)} \\
\lambda_{uh} &= \frac{h}{M-h} \cdot \frac{M-1}{M} \frac{1}{x+r} \\
&= M \cdot \frac{h-1}{M-h} \cdot \lambda_{u1}
\end{aligned}$$

The average arrival rates and the service times along with the maximum arrival rates for the hop and loop buffers are tabulated in Table 6.2 for $M = 50,\ x = 100,\ r = 6$ at 10^5 bps. These values are also given in Figure 6.10. Thus the maximum throughput at a node is the column

Table 6.2: Maximum arrival rates of buffers in LTH switching.

h	λ_1	$\overline{\chi_s}$	λ_{u1} $(pkts/sec)$	λ_{uh} $(pkts/sec)$
2	.5102 λ	.9795 λ	36.98	38.52
5	.8163 λ	.9183 λ	23.11	102.71
10	.9183 λ	.8163 λ	20.54	231.07
25	.9795 λ	.5102 λ	19.26	924.48

4 from the Table 6.2. In LTH also the throughput of a node is more than SRST.

6.3 Delay performance with varying hop sizes

At higher loads and larger hop sizes the loop traffic increases compared to hop traffic. Although packets are routed along the shortest path, a great deal of waiting is due to slower rotation of loop token. In view of this, the delay can not be more than MRMT model. More over a packet of $h1-$traffic waits at two nodes: at the sender and at the intermediate nodes. This additional waiting at the intermediate node buffer contributes some amount of additional delay. However, the delays at all loads are smaller than SRST model, due to the shortest path routing and

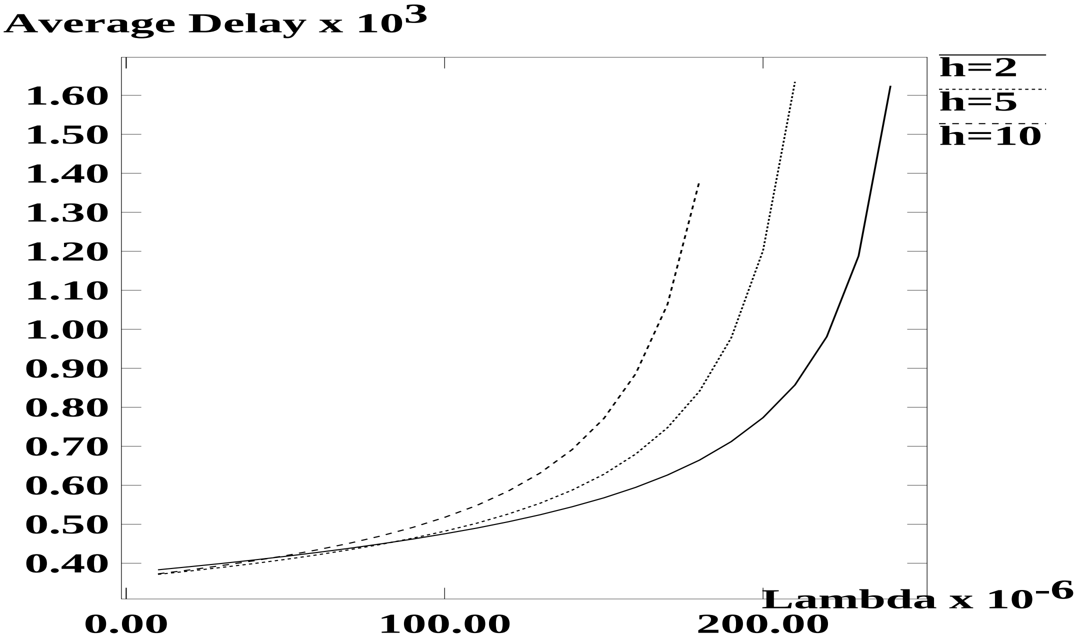


Figure 6.10: Delays in LTH showing decrease in throughput for different hop sizes (Table 6.2)

distribution of traffic over the rings. This is clear from the Figures 6.1 through 6.8.
It is of interest to see the delay performance when $g = GCD(M, h) \neq h$, such as in topologies where $1 \leq g < h$. For example in $GCD(51; +1, +6)$ and $GCD(50; +1, +7)$ cases. In the analysis of these cases, all the MAC, routing and switching conditions remain the same as the case of $GCD = h$. The main deviation in these cases is finding the fractional $\lambda_{h0}, \lambda_{01}$ and λ_{h1} values. The expressions of (5.1), (5.2) and (5.3) do not straight away hold good here. Instead we have to apply the shortest path Algorithm 3.2 to find the following from node 0:

1. the number of nodes that have only $[+h]$ links in their shortest path. Call them $h-$nodes or M_h
2. the number of nodes that have only $[+1]$ links in their shortest path. Call them $1-$nodes or M_1.
3. the number of nodes that have both $[+h]$ and $[+1]$ links in their shortest path. Call them $h1-$nodes or M_{h1}.

As the network is symmetric, the above are same from any other node also. The $h-$nodes contribute for λ_{h0}, the $1-$nodes contribute for λ_{01} and the $h1-$nodes contribute for λ_{h1} fractions. Then $\lambda_{h0}, \lambda_{01}$ and λ_{h1} are given by

$$\lambda_{h0} = \frac{M_h}{M-1} \cdot \lambda \tag{6.10}$$

$$\lambda_{01} = \frac{M_1}{M-1} \cdot \lambda \tag{6.11}$$

$$\lambda_{h1} = \frac{M_{h1}}{M-1} \cdot \lambda \tag{6.12}$$

Moreover the following results are also needed in the delay computation:

$$\begin{aligned} Number\ of\ nodes\ on\ a\ hop\ ring &= \frac{M}{g} \\ Average\ shortest\ distance\ on\ a\ hop &= \frac{M_h + 1}{2} \\ Average\ shortest\ distance\ on\ loop &= \frac{M_1 + 1}{2} \end{aligned}$$

Note that when $g = h$, we get $M_h = \frac{M}{h} - 1,\ M_1 = h - 1$, number of hop nodes are $\frac{M}{h}$, average shortest distance on a hop is $\frac{M}{h} \cdot \frac{1}{2}$ and average shortest distance on loop is $h \cdot \frac{1}{2}$ so that we get get (5.1), (5.2) and (5.3). Now continuing the delay calculations, we give in the following for the case of HTL. The case of LTH follows from this.

In equations (5.4) and (5.5) the average traffics that enter the hop and loop buffer are given for the case of $g = h$. In the present case of $g \neq h$, the corresponding average traffics λ_h and λ_1 are given by

$$\begin{aligned} \lambda_h &= \lambda_{h0} + \lambda_{h1} \\ &= \frac{M_h}{M-1}\lambda + \frac{M_{h1}}{M-1}\lambda \\ \lambda_1 &= \lambda_{01} + \lambda_{h1} \\ &= \frac{M_1}{M-1}\lambda + \frac{M_{h1}}{M-1}\lambda \end{aligned}$$

The weights of hop and loop ring traffics are similar to (6.1) and (6.2)

$$\begin{aligned} w_h &= \frac{\lambda_h}{\lambda} \\ &= \frac{M_h + M_{h1}}{M-1} \end{aligned} \tag{6.13}$$

$$\begin{aligned} w_1 &= \frac{\lambda_1}{\lambda} \\ &= \frac{M_1 + M_{h1}}{M-1} \end{aligned} \tag{6.14}$$

In Section 5.4.1, we proved that the switch over traffic is Poisson for the case of $g = h$. Similarly, the proof for the case of $g \neq h$ is also the same, except that we substitute $\frac{M}{g}$ for the number of hop nodes in place of $\frac{M}{h}$. Thus we get $\overline{P}$ and σ_P^2 similar to equations (5.16) and (5.17) that are given by

$$\begin{aligned} \overline{P} &= \frac{M}{g} \cdot \lambda_h \\ \sigma_P^2 &= (\frac{M}{g} \cdot \lambda_h)^2 [\frac{1}{\frac{M}{g}\lambda_h} - \frac{r}{1 - \frac{M}{g}\lambda_h x}] \end{aligned}$$

Similar to the approximation done for σ_P^2 in the case of $g = h$, now also we get:

$$\sigma_P^2 \approx \overline{P}$$

This indicates that P is approximately Poisson. Similar to the argument given in equation (5.20) in finding the switch over traffic X^s for the case of $g = h$, we have now for the case of $g \neq h$ the switch over traffic X^s given by

$$X^s = \frac{M_1}{M_1+1} \cdot \frac{1}{\frac{M}{g}} \cdot P$$

Thus X^s is also approximately Poisson. So χ_s is Poisson with an average rate given by, similar to equation (5.21)

$$\begin{aligned} \overline{\chi_s} &= \overline{X^s} + \lambda_{01} \\ &= \frac{M_1}{M_1+1} \cdot \frac{M_h + M_{h1}}{M-1} \cdot \lambda + \lambda_{01} \end{aligned} \quad (6.15)$$

The delays are given in the following, similar to (6.4), (6.6) in case of HTL switching. Thus we have D_h and D_1 for the case of $1 \leq g < h$:

$$D_h = \frac{M/g}{2}\left[\frac{r}{1-(M/g)\lambda_h \cdot x - (M/g)\lambda_h \cdot r} + x^2\lambda_h\right] + x + \frac{M_h+1}{2} \cdot r \quad (6.16)$$

$$D_1 = \frac{M}{2}\left[\frac{r}{1-M\overline{\chi_s} \cdot x - M\overline{\chi_s} \cdot r} + x^2\overline{\chi_s}\ \right] + x + \frac{M_1+1}{2} \cdot r, \quad (6.17)$$

where $\overline{\chi_s}$ is as given in (6.15).

In LTH switching the D_h and D_1 are similar to (6.7), (6.9):

$$D_1 = \frac{M}{2}\left[\frac{r}{1-M\lambda_1 \cdot x - M\lambda_1 \cdot r} + x^2\lambda_1\right] + x + \frac{M_1+2}{2} \cdot r \quad (6.18)$$

$$D_h = \frac{M/g}{2}\left[\frac{r}{1-(M/g)\overline{\chi_s} \cdot x - (M/g)\overline{\chi_s} \cdot r} + x^2\overline{\chi_s}\ \right] + x + \frac{M_h+1}{2} \cdot r \quad (6.19)$$

where $\overline{\chi_s}$ is in the folowing equation (6.20) and is similar to the one in (5.27).

$$\begin{aligned} \overline{\chi_s} &= \overline{X^s} + \lambda_{01} \\ &= \frac{M_h}{M_h+1} \cdot \frac{M_1 + M_{h1}}{M-1} \cdot \lambda + \lambda_{h0} \end{aligned} \quad (6.20)$$

The weighted average delay is same as (6.3) in both the switchings.

The following plots compare MCDL with SRST and MRMT for different M and h values in both the HTL and LTH switchings. In Figures 6.11, 6.12 we plot MCDL delays with $M = 120$ and $h = 2,\ 3,\ 4,\ 5,\ 6$ ($g = h$). In Figures 6.13, 6.14 we plot MCDL delays with $M = 50$ and $h = 3,\ 6,\ 7,\ 8$ ($1 \leq g < h$). Finally in Figures 6.15, 6.16 we plot MCDL delays with $M = 120$ and $h = 7,\ 8,\ 11$ ($1 \leq g < h$). It can be observed from these Figures that

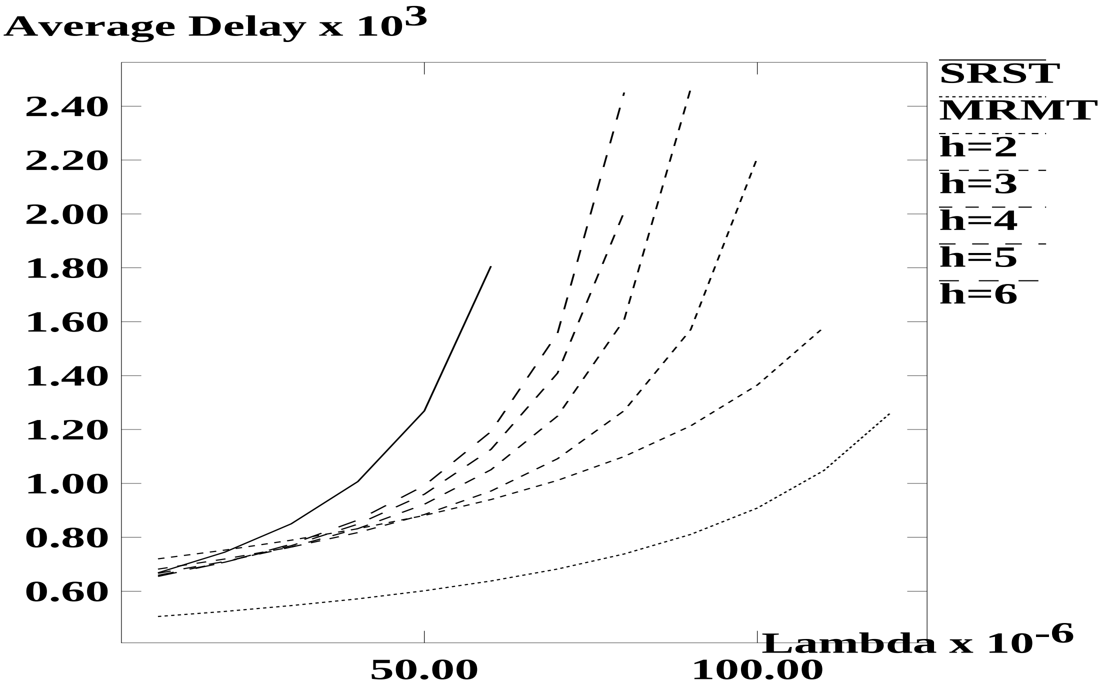


Figure 6.11: MCDL delays in HTL for M=120 and $g = h$

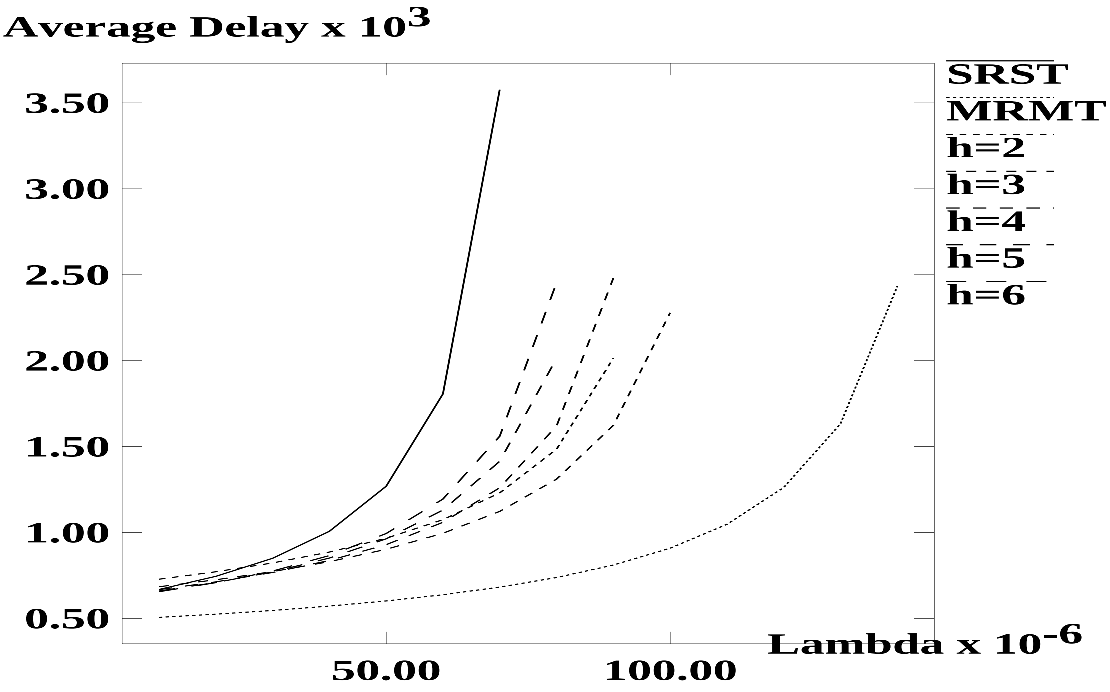


Figure 6.12: MCDL delays in LTH for M=120 and $g = h$

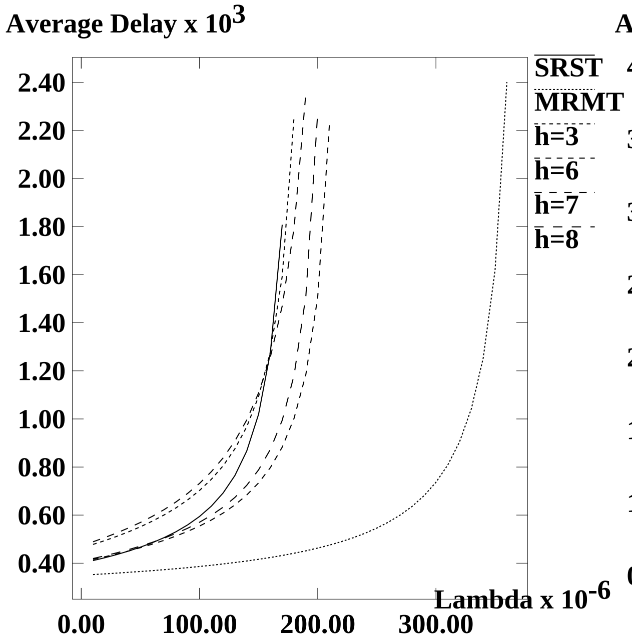


Figure 6.13: MCDL delays in HTL for M=50 when $1 \leq g < h$

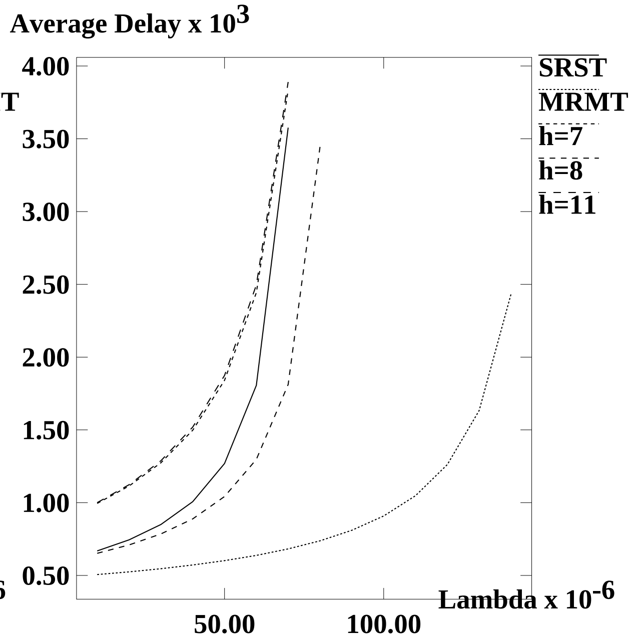


Figure 6.15: MCDL delays in HTL for M=120 when $1 \leq g < h$

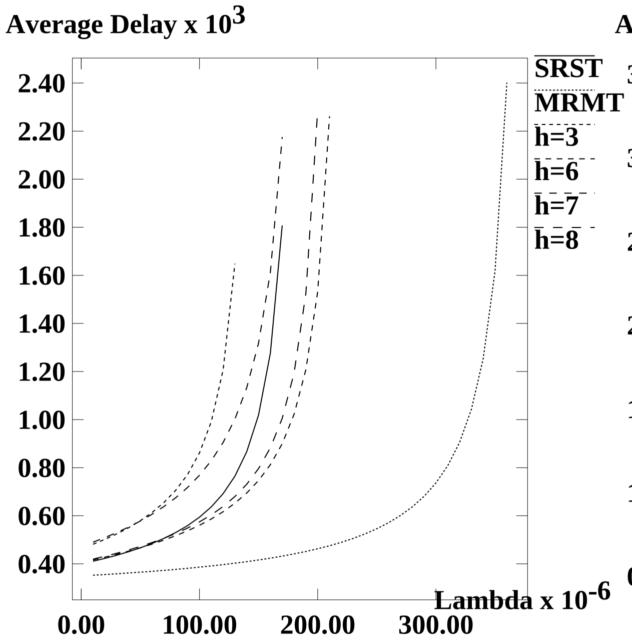


Figure 6.14: MCDL delays in LTH for M=50 when $1 \leq g < h$

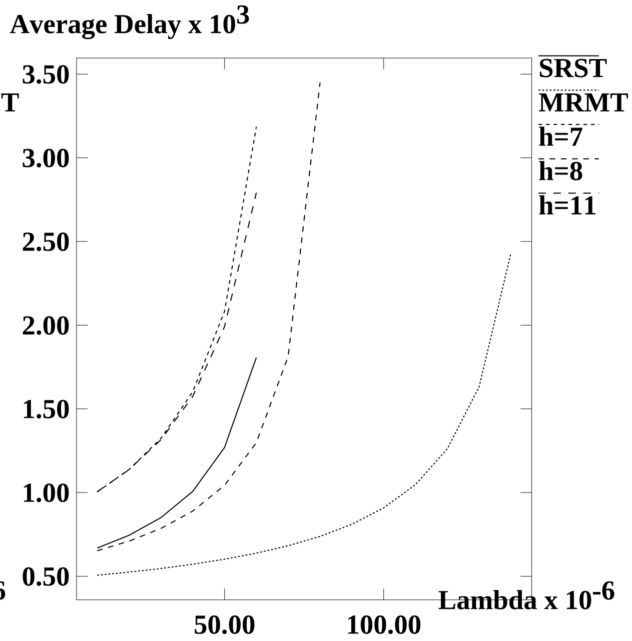


Figure 6.16: MCDL delays in LTH for M=120 when $1 \leq g < h$

when $g = 1$ the delays are more than SRST model, for example in $M = 120$, $h = 7, 11$ (Figures 6.15 and 6.16) and $M = 50$, $h = 3, 7$ (Figures 6.13 and 6.14). It is due to the following reasons:

1. When $g = 1$, the number of nodes on the hop ring are same as the loop ring. Thus hop ring is a parallel ring to loop ring like MRMT with $B = 2$.

2. In view of the above, the delay at the two buffers is the dominating component in spite of the shortest path used.

when $1 < g < h$, the delays are less than SRST but more than the case of $g = h$. It is due to the following reasons:

1. The number of nodes on the hop ring are more than the case of $g = h$, but less than the case of $g = 1$.

2. Hence the shortest path dominates the delay than the waiting at two buffers, but not as much as the case of $g = h$.

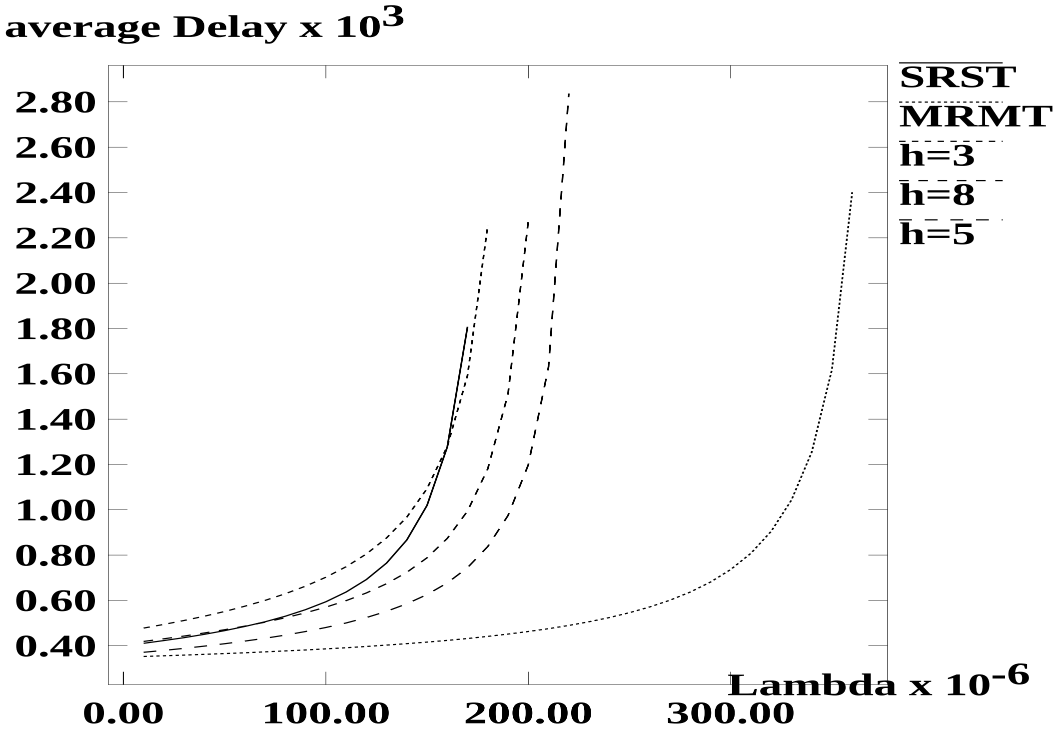


Figure 6.17: Effect of GCD on delays in HTL M=50 for different hop sizes

Thus the delay can be observed to decrease in the following order: $MCDL$ with $g = 1$, $SRST$, $MCDL$ with $1 < g < h$, $MCDL$ with $g = h$ followed by $MRMT$. In Figure 6.17, we have

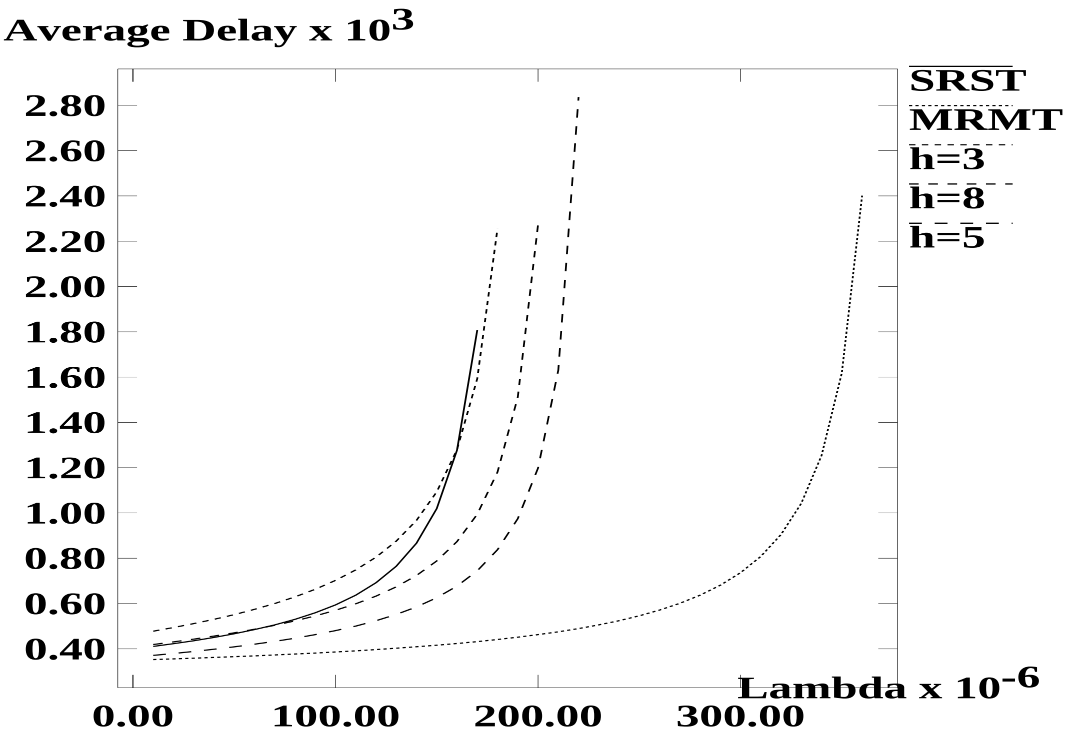


Figure 6.18: Effect of GCD on delays in LTH M=50 for different hop sizes

taken $M = 50$, $SRST$, $MRMT$ with ($B = 2$), $MCDL$ with $h = 3$ ($g = 1$), $MCDL$ with $h = 8$ ($g = 2$) and $MCDL$ with $h = 5$ ($g = h$). Similar results can be seen in LTH switching also in Figure 6.18.

6.4 Delay performance with varying packet length

In a computer network two types of message lengths are of interest in the delay analysis. They are *fixed length* packets and *variable length* packets. In file transfer applications the packet lengths are generally of fixed size. But in interactive mode applications the packet lengths are of variable in size and also are short. So far we have considered fixed length packets in our analysis. In the following we see the effect of variable length packets in the delay performance. Variable length packets are generally analysed by taking the packet lengths from a probability distribution [Kle75], usually the *exponential distribution*. It is the most suitable distribution for this purpose, since it has memory less property [Med94]. If lengths of packets are chosen from the exponential distribution, then successive packet lengths are independent as explained in the following.

The probability distribution of exponential distribution is given by

$$\begin{aligned} F(x) &= Pr\ [\ X \leq x\ ] \\ &= 1 - e^{\theta x} \end{aligned} \tag{6.21}$$

where $F(x)$ is the distribution function for the random variable x (i.e., packet length). The θ is the parameter of the distribution. The mean and variance of x are

$$\begin{aligned} \overline{x} &= \frac{1}{\theta} \\ \sigma_x^2 &= \frac{1}{\theta^2} \\ &= (\overline{x})^2 \end{aligned}$$

The $\overline{T}$, σ_T^2 and ρ are derived in [SS85] for SRST model for exponentially distributed packet lengths. They are

$$\begin{aligned} \overline{T} &= M\rho\overline{x} + Mr \\ \sigma_T^2 &= M\rho(2-\rho)\overline{x}^2 \\ \rho &= \lambda \cdot \overline{T} \end{aligned}$$

Then the waiting time is obtained by substituting the above in the waiting time expression given in (C.6),

$$W = \frac{M}{2}[\frac{r}{1 - M \cdot \lambda \cdot \overline{x} - M \cdot \lambda \cdot r} + \overline{x}^2\lambda\frac{2 - 2M\lambda\overline{x} - Mr\lambda}{1 - M\lambda\overline{x} - Mr\lambda}] \tag{6.22}$$

$$D = W + \overline{x} + \frac{M}{2} \cdot r \tag{6.23}$$

where $\overline{x}$ is the mean of the exponentially distributed packet length. It is observed in [SS85] that the difference in the packet delays between fixed length and exponentially distributed packet lengths is negligible except at high traffic and large value of M (refer Figures 6.19). In our MCDL model with $g = h$, the delays are given below for HTL switching (refer the corresponding delays for MCDL model HTL switching fixed packet length given in (6.4) and (6.6))

$$\begin{aligned} D_h &= \frac{M/h}{2}[\frac{r}{1 - (M/h)\lambda_h\overline{x} - (M/h)\lambda_h r} + \overline{x}^2\lambda_h\frac{2 - 2(M/h)\lambda_h\overline{x} - (M/h)r\lambda_h}{1 - (M/h)\lambda_h\overline{x} - (M/h)r\lambda_h}] \\ &\quad + \overline{x} + \frac{M}{h} \cdot r \cdot \frac{1}{2} \end{aligned} \tag{6.24}$$

$$D_1 = \frac{M}{2}[\frac{r}{1 - M\overline{\chi_s}\overline{x} - M\overline{\chi_s}r} + \overline{x}^2\overline{\chi_s}\frac{2 - 2M\overline{\chi_s}\overline{x} - Mr\overline{\chi_s}}{1 - M\overline{\chi_s}\overline{x} - Mr\overline{\chi_s}}\ ] + \overline{x} + \frac{h}{2} \cdot r, \tag{6.25}$$

$$D_a = w_h D_h + w_1 D_1$$

Similarly, for LTH switching (refer the corresponding delays for MCDL model LTH switching fixed packet length given in (6.9 and (6.7))

$$D_1 = \frac{M}{2}[\frac{r}{1 - M\lambda_1\overline{x} - M\lambda_1 r} + \overline{x}^2\lambda_1\frac{2 - 2M\lambda_1\overline{x} - Mr\lambda_1}{1 - M\lambda_1\overline{x} - Mr\lambda_1}] + \overline{x} + h \cdot r \cdot \frac{1}{2} \tag{6.26}$$

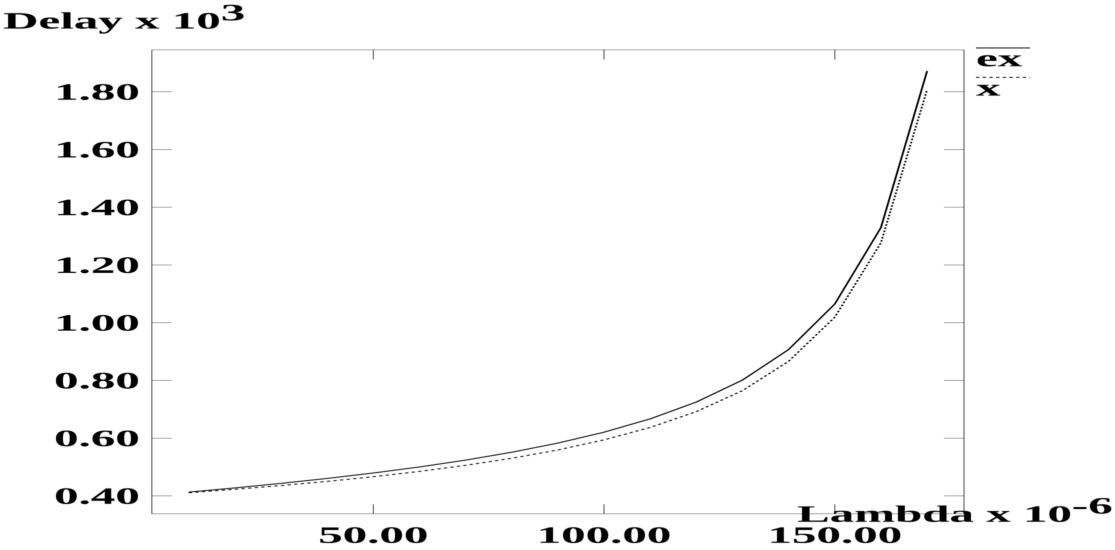


Figure 6.19: Effect on delay for fixed and exponentially distributed packet length in SRST

$$
\begin{aligned}
D_h &= \frac{(M/h)}{2}\left[\frac{r}{1-(M/h)\overline{\chi_s x}-(M/h)\overline{\chi_s}r} + \overline{x}^2\overline{\chi_s}\frac{2-2(M/h)\overline{\chi_s x}-(M/h)r\overline{\chi_s}}{1-(M/h)\overline{\chi_s x}-(M/h)r\overline{\chi_s}}\right] \\
&\quad +\overline{x} + \frac{M}{h}\cdot\frac{1}{2}\cdot r, \qquad (6.27) \\
D_a &= w_h D_h + w_1 D_1
\end{aligned}
$$

The delays are plotted in Figures 6.20 through 6.22 for HTL switching, $M = 50$ and $h = 2, 5, 10$ for the fixed (x in Figure) and exponentially (ex in Figure) distributed packet lengths.

6.5 Delay performance in NOS

It is of interest to study the the third type of routing explained in Section 5.2 *i.e.*, no-switching. In this case, the $h1-$traffic is not switched at the intermediate node. Instead, it goes to the destination straight from the source along the loop token. Hence $h1-$traffic is to be routed only on the loop ring. The other traffics *i.e.*, $h0$ and 01 are to be routed on the hop and loop rings respectively. In such case, when $g = h$

$$
\begin{aligned}
\lambda_h &= \lambda_{h0} \\
&= \frac{\frac{M}{h}-1}{M-1}\lambda \\
\lambda_1 &= \lambda_{01} + \lambda_{h1} \\
&= \frac{\frac{M}{h}(h-1)}{M-1}\lambda
\end{aligned}
$$

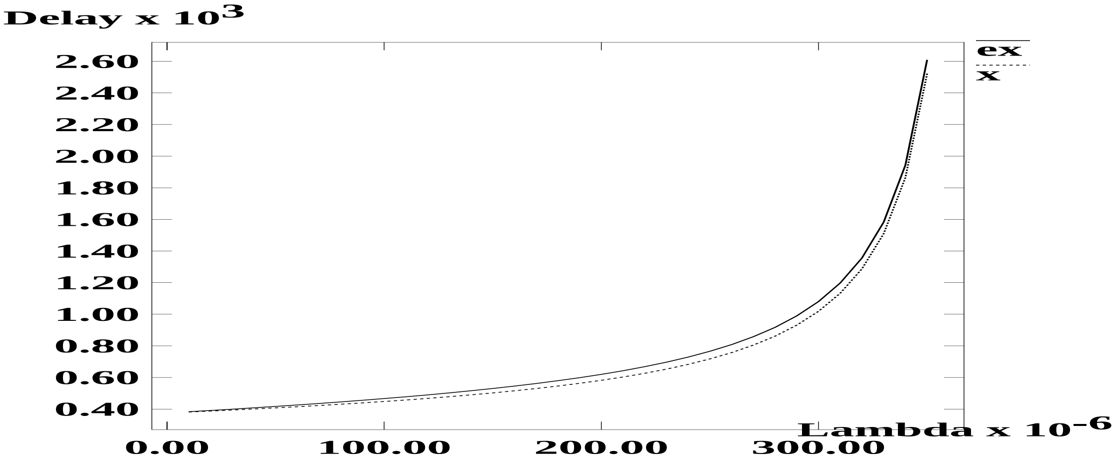


Figure 6.20: Effect on delay for fixed and exponentially distributed packet lengths in MCDL, h=2, HTL

For the case of $g \neq h$ we have

$$\begin{aligned}\lambda_h &= \frac{M_h}{M-1}\lambda \\ \lambda_1 &= \frac{M-1-M_h}{M-1}\lambda\end{aligned}$$

However, the w_h and w_1 are

$$\begin{aligned}w_h &= \frac{\lambda_h}{\lambda} \\ w_1 &= \frac{\lambda_1}{\lambda}\end{aligned}$$

Correspondingly the delays of hop and loop rings for the case of $g = h$ are

$$\begin{aligned}D_h &= \frac{(M/h)}{2}[\frac{r}{1-(M/h)\lambda_h x-(M/h)\lambda_h r}+x^2\lambda_h\ ]+x+\frac{M/h}{2}\cdot r \\ D_1 &= \frac{M}{2}[\frac{r}{1-M\lambda_1 x-M\lambda_1 r}+x^2\lambda_1\ ]+x+\frac{M}{2}\cdot r \\ D_a &= w_h D_h + w_1 D_1\end{aligned}$$

The corresponding delay equations for $g \neq h$ are same as above except that the number of nodes on hop ring is $\frac{M}{g}$ and the last term *i.e.* , the average propagation time for the hop ring. It is $\frac{M_h+1}{2} \cdot r$ for a hop ring. We plot the weighted average delay in Figure 6.23 to 6.26 for $M = 50$, $h = 2, 5, 6, 7$ and compare with HTL switching. It is clear from these figures that the delays under no-switching are less than the switching cases. Moreover, these delays are less than the MRMT also under small hop sizes such as h=2, when M=50.

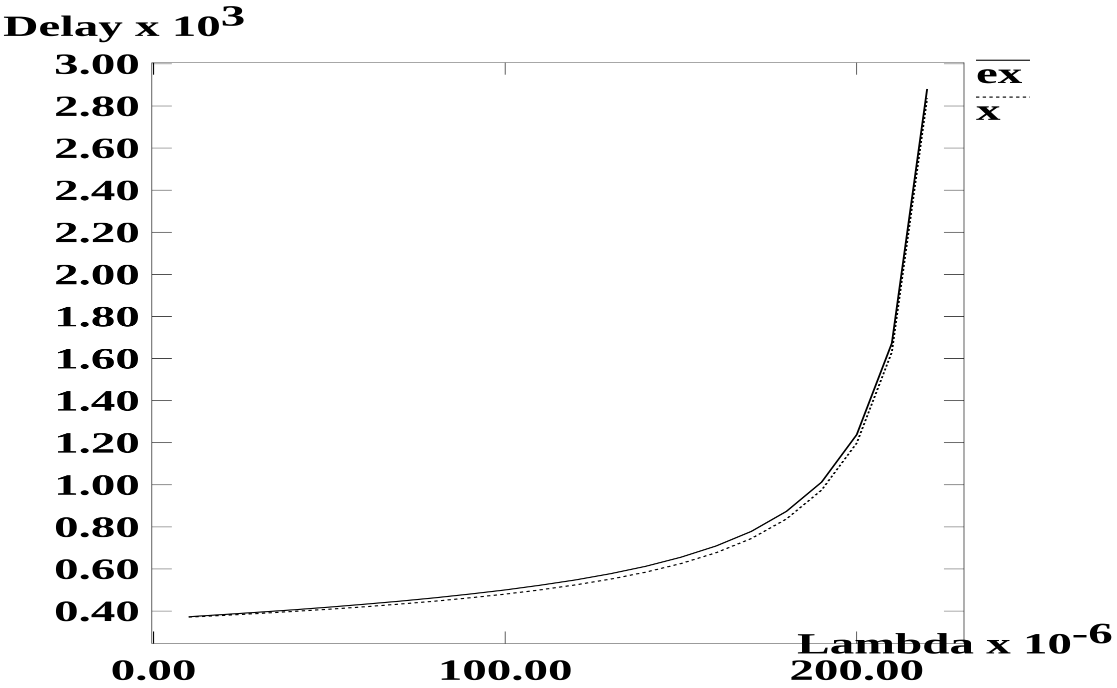


Figure 6.21: Effect on delay for fixed and exponentially distributed packet lengths in MCDL, h=5, HTL

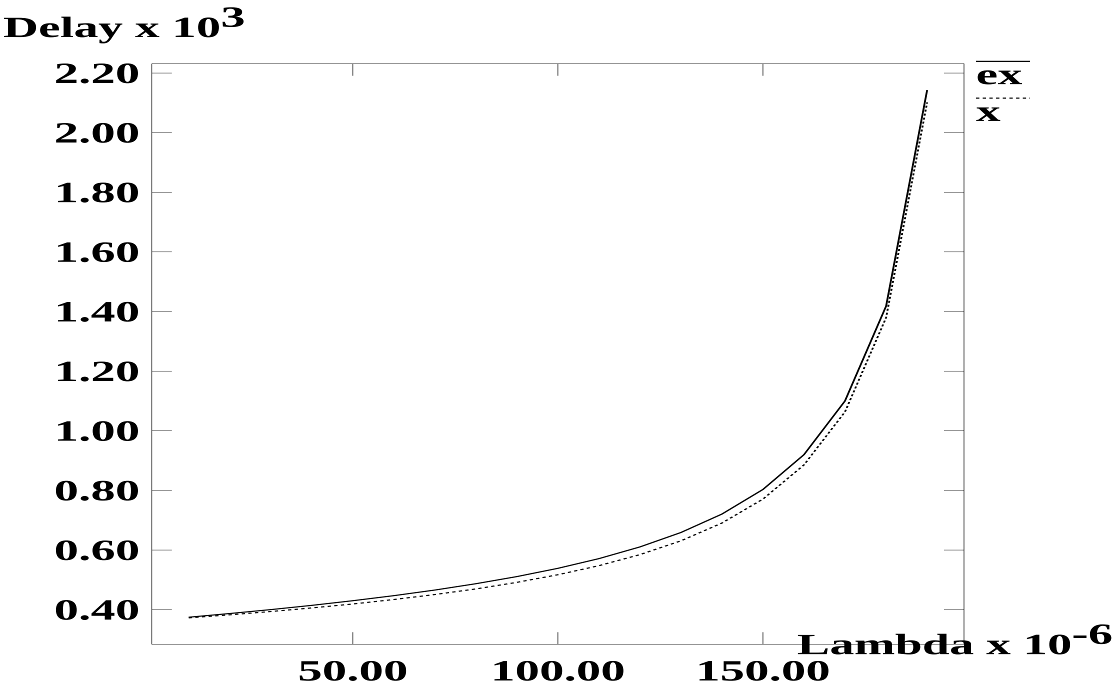


Figure 6.22: Effect on delay for fixed and exponentially distributed packet length in MCDL, h=10, HTL

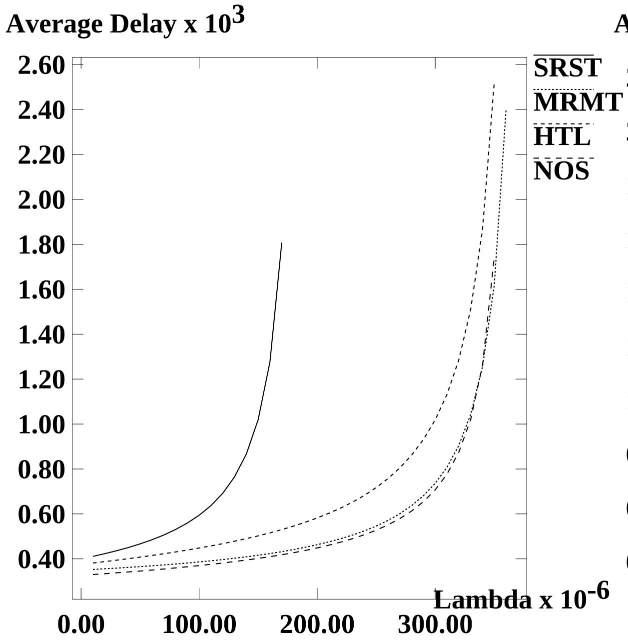


Figure 6.23: Comparison of delays in HTL h=2 and NOS routings

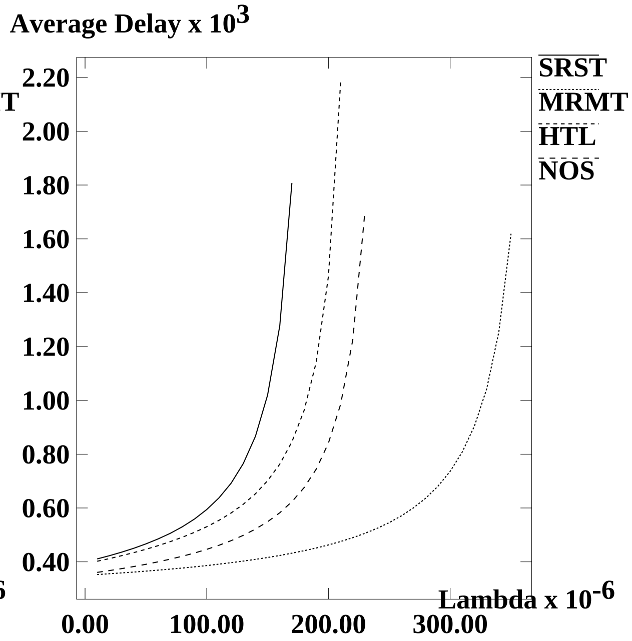


Figure 6.25: Comparison of delays in HTL h=6 and NOS routings

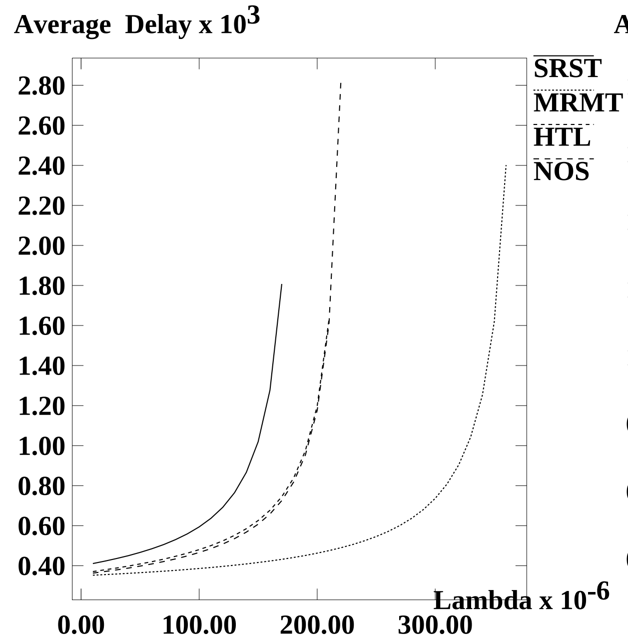


Figure 6.24: Comparison of delays in HTL h=5 and NOS routings

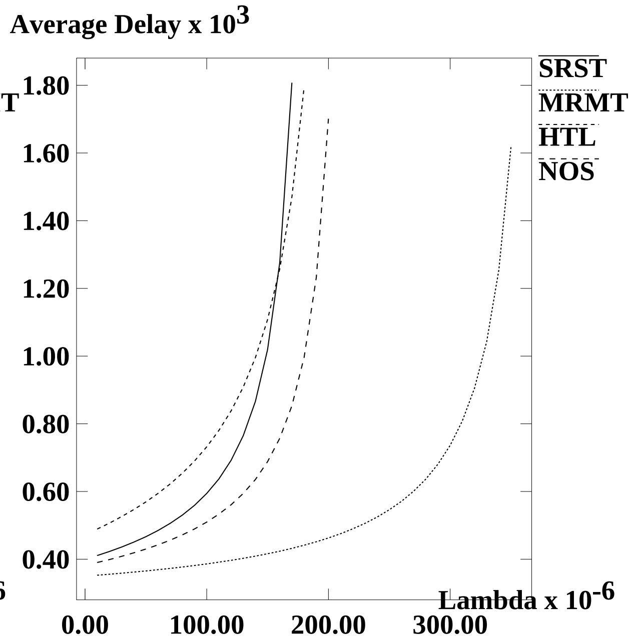


Figure 6.26: Comparison of delays in HTL h=7 and NOS routings

6.6 Comparison of MCDL with SRST and MRMT

The comparison of delays among SRST,MRMT and MCDL with different hop sizes and switching types show the following observations

- Delays of MRMT are less than SRST due to the multiplicity of channels, access points and tokens.
- The delays of MCDL when $g \neq 1$ are also less than SRST by the same argument that MCDL also has multiple channels and tokens as servers. But they are more than MRMT due to the waiting time at two buffers. Thus the delays of MCDL when $g \neq 1$ lie between the SRST and MRMT.
- In both the switching types of MCDL *viz*, HTL and LTH, the delays are same. It is due to the symmetry in the topology and routing.
- In the case of NOS in MCDL, the delays are smaller than HTL and LTH switching, because the packets no more wait in two buffers. However these delays are still more than MRMT except for very small hop sizes such as $h = 2$ in $M = 50$. The reason as why the delays in NOS are more than MRMT in spite of no waiting at two buffers is that as the packets arrive a host node in MCDL they are classified based on its destination address and are transmitted on the corresponding channel (hop or loop ring) only when the requisite token on this channel arrives. Until such time the packets wait in the buffers. But in MRMT, the packets that arrive a host are transmitted when any of the tokens from the connected channels (the B rings) is received.
- We compare the delays for different hop sizes when $g = h$. It is seen that as hop size increases the delays increase. For small hop sizes (say $h = 2$) the delays are close to MRMT as the topology almost resembles that of MRMT in the number of nodes on the hop and loop rings and the respective traffics on them. As the hop size increases the number of nodes that come on hop ring decreases and consequently the loop traffic increases. Thus the loop token rotation time increases and this limits the overall delay performance. Thus we see an increase in the average delay for larger hop sizes. However we still get an advantage over the SRST model.
- When the $g = 1$, the number of hop and loop nodes are same, the delay at two buffers contributes more for the average delay than any gain derived from the shortest path transmission.
- The effect of packet length in both MRMT and MCDL is same. As the packet length increases the token hold time at a station also increases. Thus delaying the release of token

to the next station. This increases the token rotation time. In turn the packet average delays increase. We plotted the average delay versus λ for different packet sizes in Figure 6.27 and 6.28 *i.e*, $x = 100, 150, 200, 250, 300, 350$, $M = 50$, $r = 6$ HTL switching and $h = 2$. Note that the total round trip propagation time for loop ring is $R_1 = 50 \cdot 6 = 300$ and the one for hop ring is $R_h = \frac{50}{2} \cdot 6 = 300$.

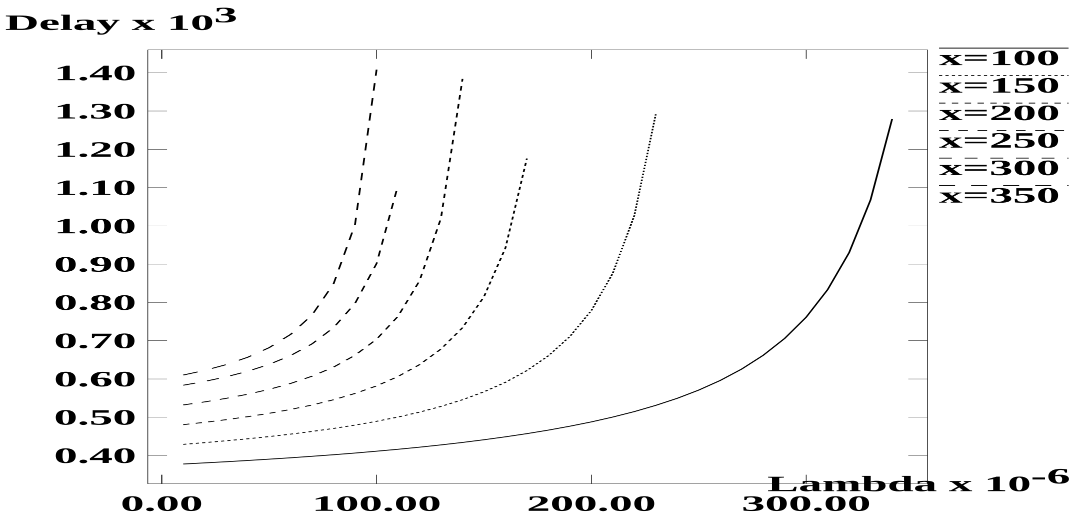


Figure 6.27: Effect of packet length on delay in MRMT, B=2

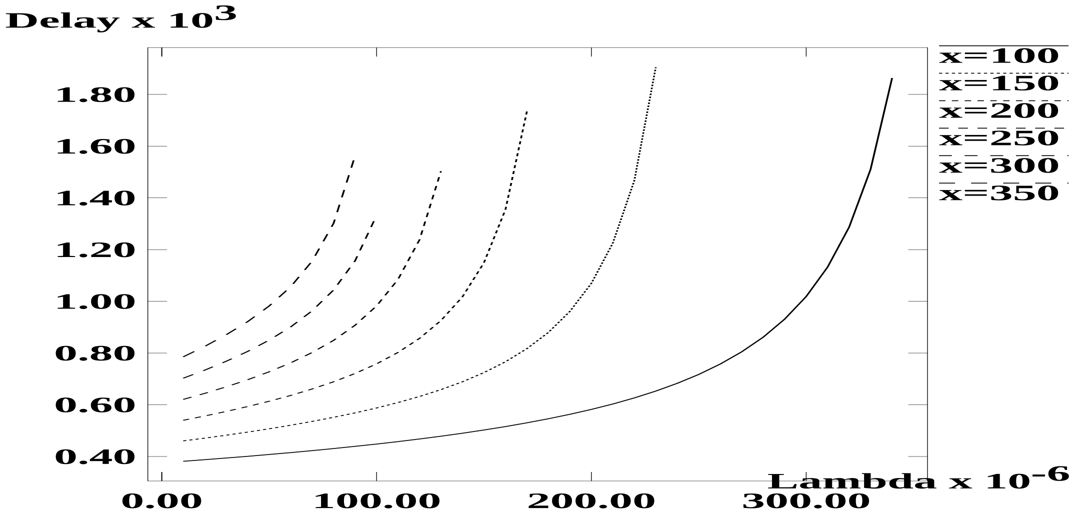


Figure 6.28: Effect of packet length on delay in MCDL, h=2, hop to loop

Chapter 7

DISCRETE EVENT SIMULATOR FOR MCDL

A program was developed to simulate the $MCDL$ multiple token network. This was designed to measure packet waiting time and packet delays and compare it with the analytical results in Chapter 5 and verify the validity of assumptions made therein.

7.1 Assumptions in simulation

The simulations are under the following assumptions:

- Non-exhaustive or 1-limited service policy
- Both fixed and exponentially distributed packet lengths are considered
- Poisson arrival rate with an average rate of λ packets per sec
- Statistically identical stations
- Uniform distribution of packet destinations
- Source removal of packets after a complete cycle
- Infinite length of input buffers at all stations to receive packets
- Two transmit buffers the *h-buffer* and *1-buffer* per station are used
- FDDI token ring channel access scheme

To examine the parameters that control the delay, three different configurations were simulated: *hop to loop switching*, *loop to hop switching* and *no switching*. Simulations were run [Sha99] with different values of number of stations, hop sizes and different station-to-station propagation delays. All timing measurements are done at 10^5 *bps* and expressed in bit-time intervals (*i.e*, a bit-time interval= $\frac{1}{bit\ rate}$). In order to obtain steady state and consistent results from simulation around 200 to 400 thousand bit-time intervals are neglected initially before taking any readings. It was observed that 20 to 40 million bit-time intervals would permit accurate results within 20 packets per sec(100 bit packet size) load range [Sha99]. An arrival rate of more than 20 packets per sec causes overflow of input buffers ($\rho \approx 1$) at 10^5 bits per sec bit rate.

7.2 Simulation results

Simulations were run and the following results are obtained.

1. Number of packets carried by the hop (loop) token, delivered at a loop (hop) node, switched at loop (hop) node , their mean and variances in both the HTL and LTH switching are obtained from the simulation for different values of arrival rates.

2. Delay of packets for various number of stations at different hop sizes such as: (i) $GCD(M, h) = h$, (ii) $GCD(M, h) = 1$ and (iii) $1 < GCD(M, h) < h$ in all the three switchings are obtained with the simulation.

3. In exponentially distributed packet length (6.21) simulation, each time a packet is generated its length is selected from the exponential distribution whose mean is 100 bits. Using this packet length the mean packet delays are obtained.

7.3 Discussion of simulation results

In what follows, a comparison of simulation results with the analytic results of Chapters 5 and 6 is given [CBS98, CB99b].

In Tables 7.1 and 7.2, the simulation results for the mean number of token rotations ($\overline{n}$) and average time for each token rotation ($\overline{T_h}$ and $\overline{T_1}$) for the HTL and LTH switchings at different traffic arrival rates are given. The corresponding analytic results are from the expressions derived in Chapter 5 equations (5.13) (5.12) (5.23) and (5.22) and given in Tables 7.1 and 7.2. An examination of them concludes that analysis and simulations tally for $\overline{n}$, $\overline{T_h}$ and $\overline{T_1}$ in both the switchings.

Table 7.1: Analysis and simulation of $\overline{n}$ and $\overline{T_h}$ in HTL switching

$\lambda(10^{-6})$	hop	Ana($\overline{n}$)	Sim($\overline{n}$)	Ana($\overline{T_h}$)	Sim($\overline{T_h}$)
10	2	650	650	153	153
50	2	584	585	170	170
50	5	1601	1590	62	62
50	10	3250	3265	30	30
100	2	520	503	192	198
100	5	1532	1513	65	66
100	10	3200	3197	31	31

Table 7.2: Analysis and simulation of $\overline{n}$ and $\overline{T_h}$ in LTH switching

$\lambda(10^{-6})$	hop	Ana($\overline{n}$)	Sim($\overline{n}$)	Ana($\overline{T_h}$)	Sim($\overline{T_h}$)
60	2	282	282	354	354
90	2	256	255	389	386
150	2	205	202	485	480
180	2	180	177	554	550
60	5	251	251	397	395
120	5	170	169	588	585
180	5	88	86	1130	1120
90	10	195	195	511	511
120	10	149	149	668	668
150	10	103	102	963	960
180	10	757	755	1729	1709

In Tables 7.3 and 7.4, the simulation results for mean and variance of number of packets carried P are given for HTL and LTH switchings at different traffic arrival rates. The corresponding analytic results are from the expressions derived in Chapter 5 equations (5.16), (5.17) for HTL switching and (5.24), (5.25) for LTH switching respectively. The purpose of obtaining simulations for the mean and variance of P is to study the statistical distribution of P and the packets that are input to the loop (or hop) buffers in the corresponding switchings. In Chapter 5 equations (5.19) and (5.26) prove that they are Poisson in both the switchings. We now verify this from simulations given in Tables 7.3 and 7.4. An examination of the simulations indicate that the mean and variance of P are equal. Hence the assumption of Poisson switch over traffic is also tallied. Now we proceed to obtain the packet waiting times in hop and loop buffers, average delay for the three types of switchings by simulations. The corresponding waiting times from the analysis are found in Chapter 6 equations (6.4) and (6.6) for hop and loop in HTL switching. Similarly these waiting times for in LTH switching are from equations

(6.9) and (6.7). Note that average delay is given in equation (6.3). The hop, loop and average delays for analysis and simulation are given in Figures 7.1, 7.2 and 7.3 for $h = 5$ $(GCD = h)$, $h = 6$ $(GCD < h)$ and $h = 7$ $(GCD = 1)$ with $M = 50$ stations. For each of the hop size we have simulated for HTL, LTH and NOS routing. The graphs of analysis given in Chapter 6 are repeated here to compare with the simulation results.

Table 7.3: analysis and simulation of $\overline{P}$ and σ_P^2 in HTL switching

$\lambda(10^{-6})$	hop	Ana($\overline{P}$)	Sim($\overline{P}$)	Ana(σ_P^2)	Sim(σ_P^2)
140	2	342	337	332	325
220	2	538	529	501	524
300	2	734	718	612	626
350	2	857	836	548	691
100	5	183	179	181	182
150	5	137	136	136	125
200	5	183	179	181	182
200	10	81	79	81	90
300	10	122	120	121	112
400	10	163	159	161	160

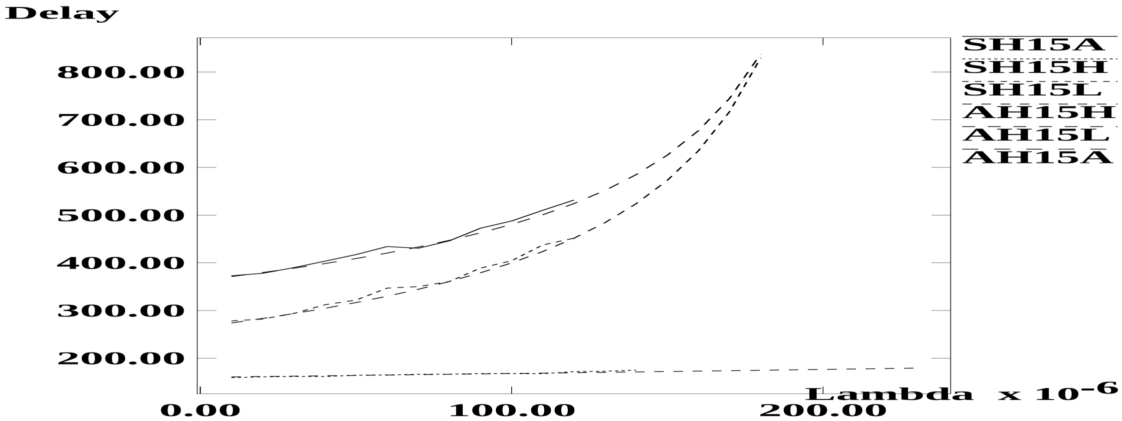
Table 7.4: analysis and simulation of $\overline{P}$ and σ_P^2 in LTH switching

$\lambda(10^{-6})$	hop	Ana($\overline{P}$)	Sim($\overline{P}$)	Ana(σ_P^2)	Sim(σ_P^2)
60	2	153	153	151	151
90	2	229	233	225	229
150	2	382	391	368	391
180	2	459	468	435	469
60	5	244	223	240	223
90	5	367	365	354	348
120	5	489	492	461	531
180	5	734	740	612	622
90	10	413	414	395	370
120	10	163	159	161	160
150	10	691	688	579	590
180	10	826	832	890	832

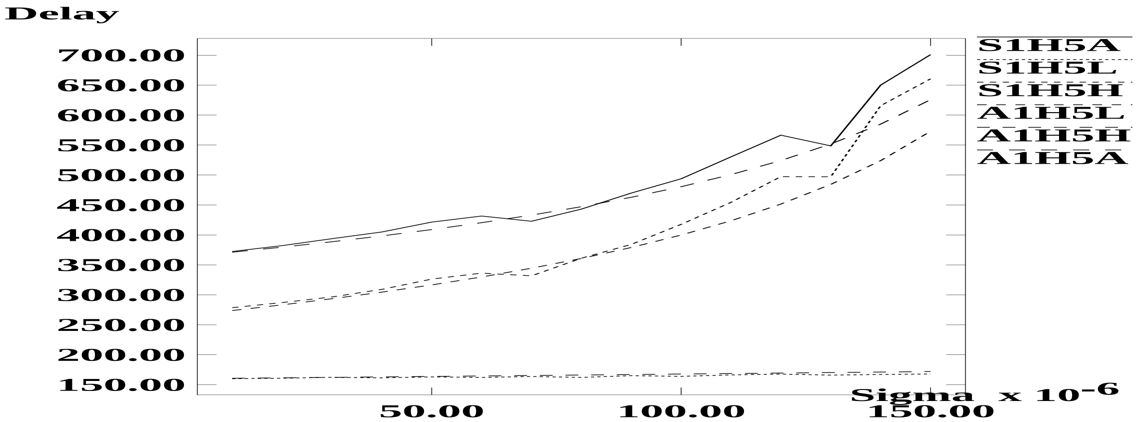
In exponentially distributed packet lengths, the hop, loop and average delays of simulation are compared with the analytic results obtained in Chapter 6 equations (6.24), (6.25) for HTL and similarly equations (6.27) and (6.26) for LTH switching. The corresponding delays are plotted in Figure 7.4 for h=2,5 and 10.

The graphs presented here are identified with a 5 character *symbol*. The first character in the symbol represents simulation (by S) or analysis (by A). The next two characters indicates switching type such as HTL (by $H1$) or LTH (by $1H$) or *no_switching* (by NO). The fourth character of the symbol is a digit signifying the hop size (by 5 or 6 or 7 in the present case). The last character is for the delay parameter such as hop delay (by H) or loop delay (by L) or average delay (by A). For example the symbol $S1H6L$ indicates loop delay simulation in LTH model for hop size as 6. Similarly $ANO7A$ indicates average delay analytic result in *no_switching* case for hop size 7.

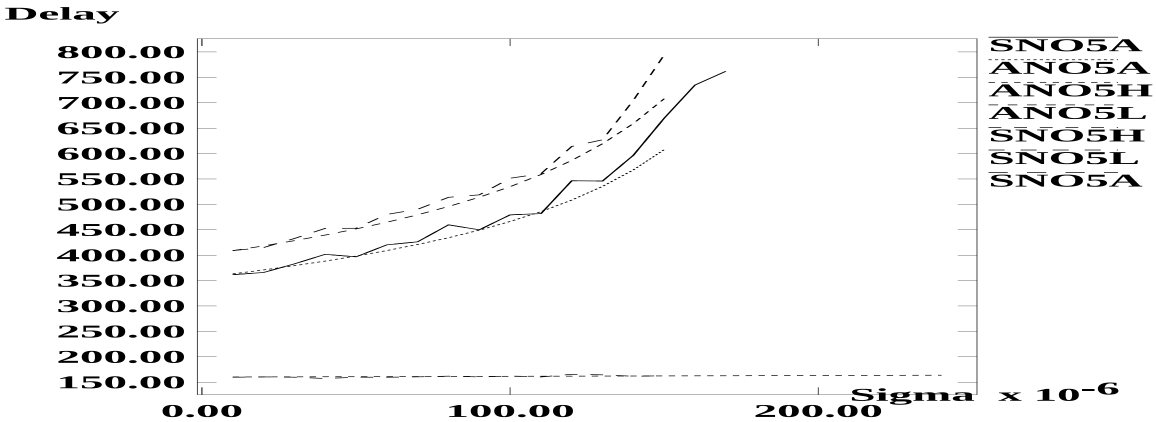
The *x-axis* (designated by λ), in the graphs represent the load (average arrival rate of packets) in *pkts/bit time*. At 10^5 *bps*, λ of 0.00015 packets per bit time (or 150×10^{-6}) is equal to 15 packets per sec. Similarly the *y-axis* designates delay in bit times. For example a delay of $2.40 \times 10^3 = 2400$ bit times is equal to 24 *msec* at 10^5 *bps*.



(a). Hop to loop (HTL)

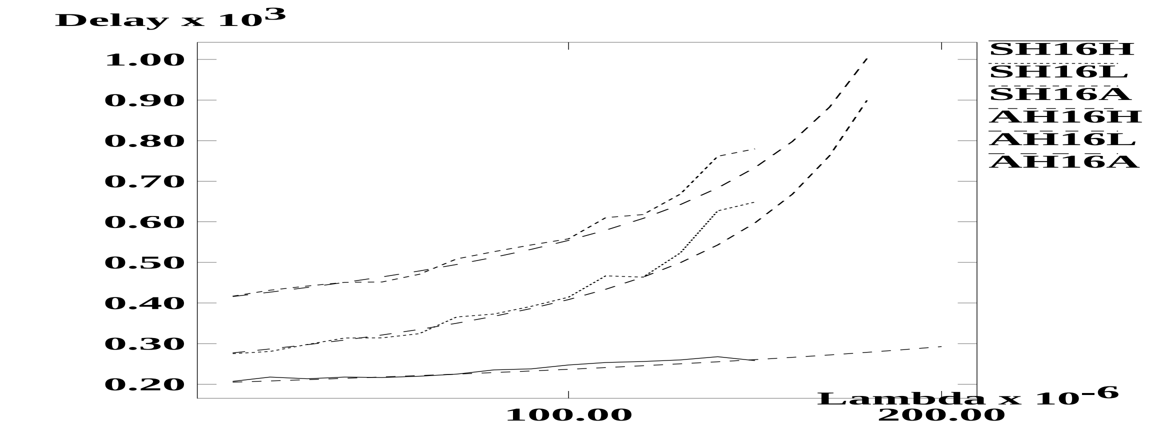


(a). Loop to hop (LTH)

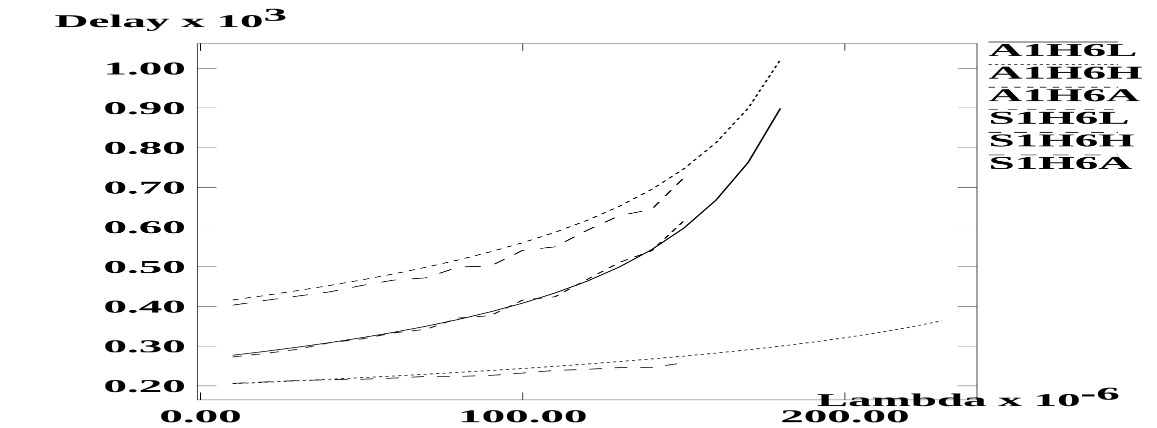


(a). No-switching (NOS)

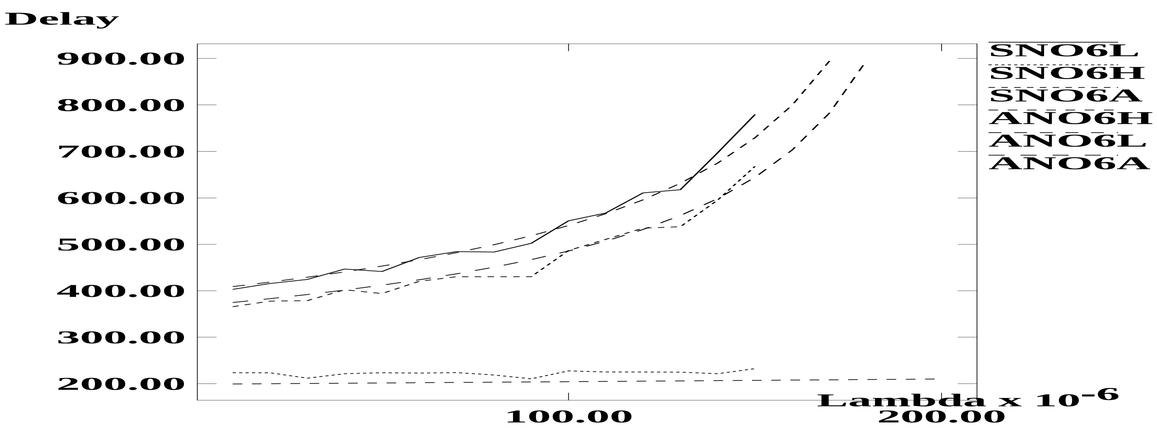
Figure 7.1: Simulation and analysis of hop, loop and avg delays in HTL, LTH and NOS for MCDL h=5



(a). Hop to loop (HTL)

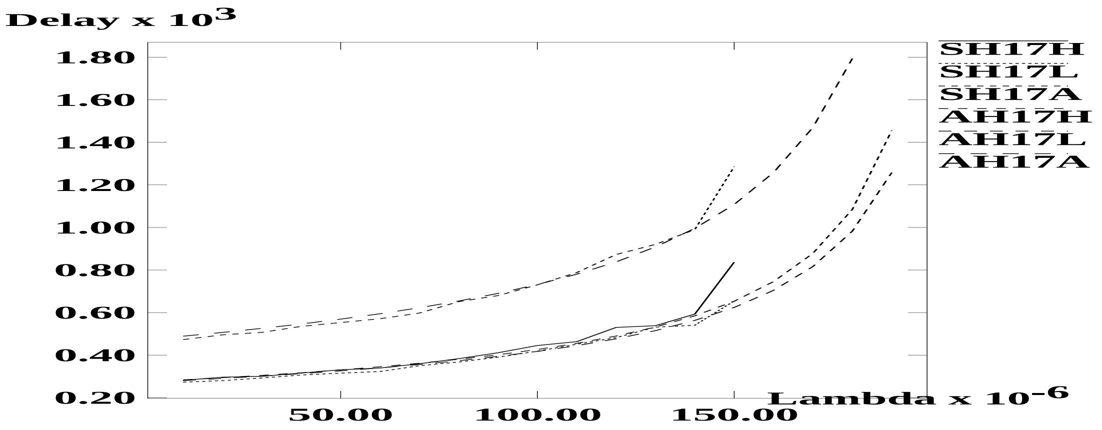


(a). Loop to hop (LTH)

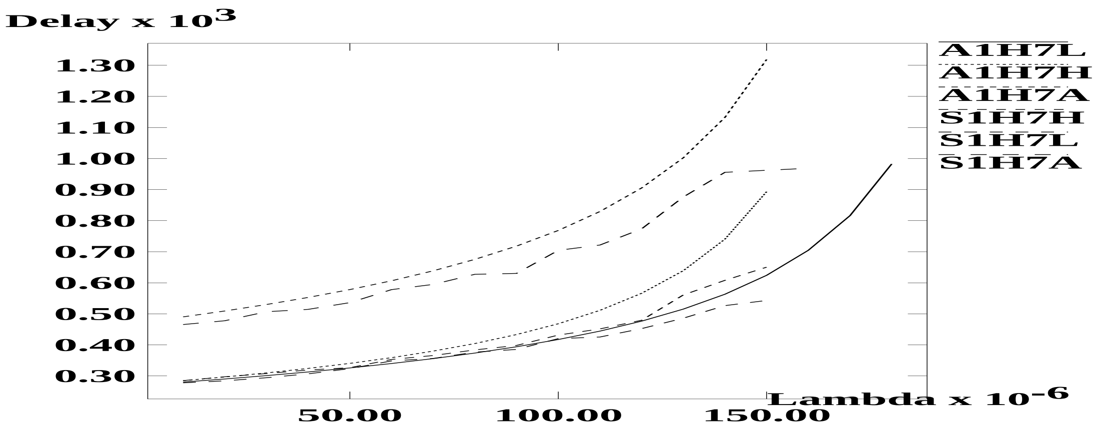


(a). No-switching (NOS)

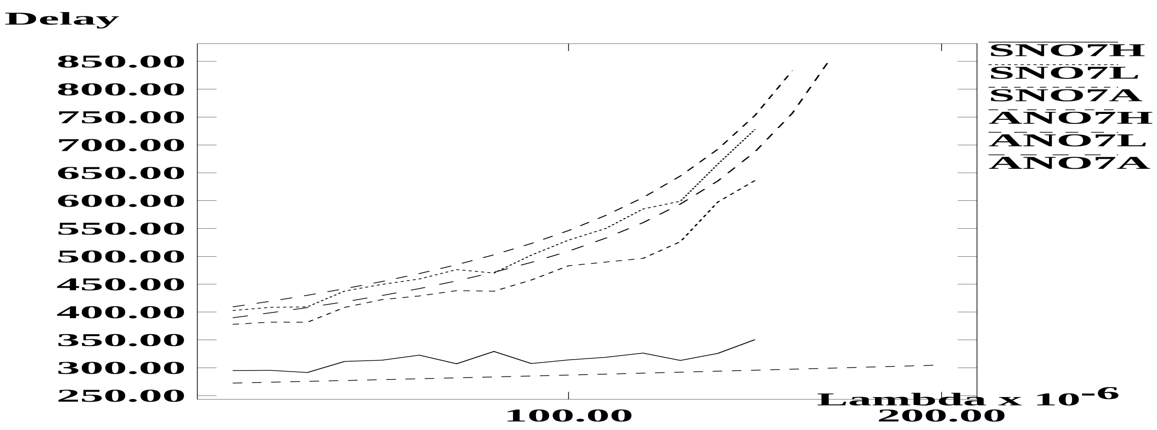
Figure 7.2: Simulation and analysis of hop, loop and avg delays in HTL, LTH and NOS for MCDL h=6



(a). Hop to loop (HTL)

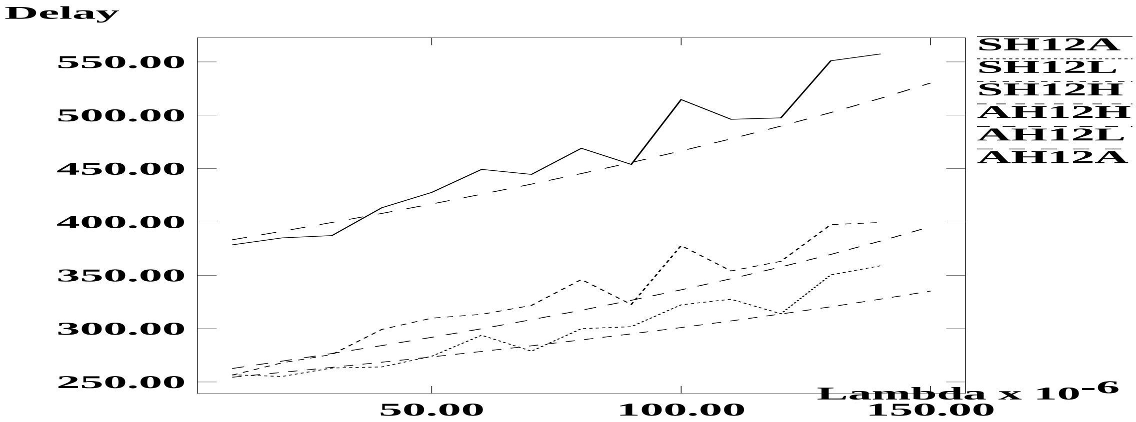


(a). Loop to hop (LTH)

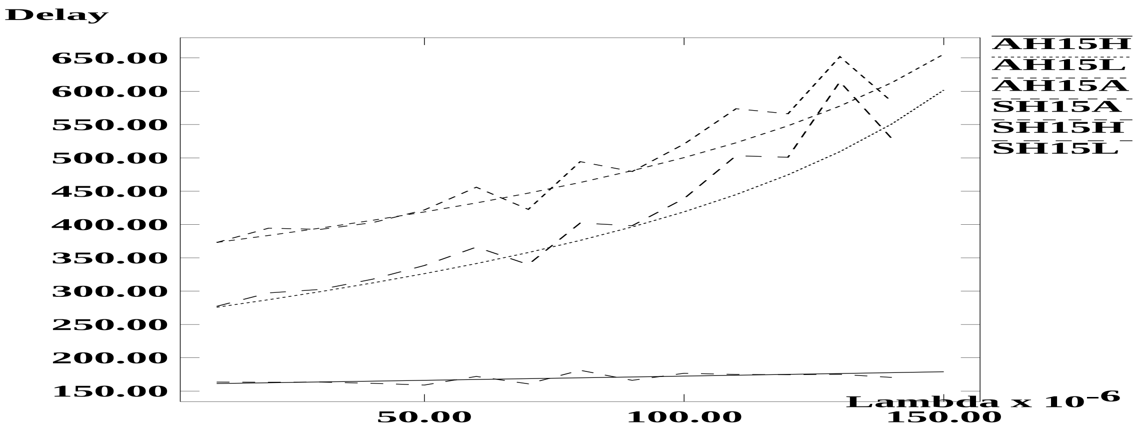


(a). No-switching (NOS)

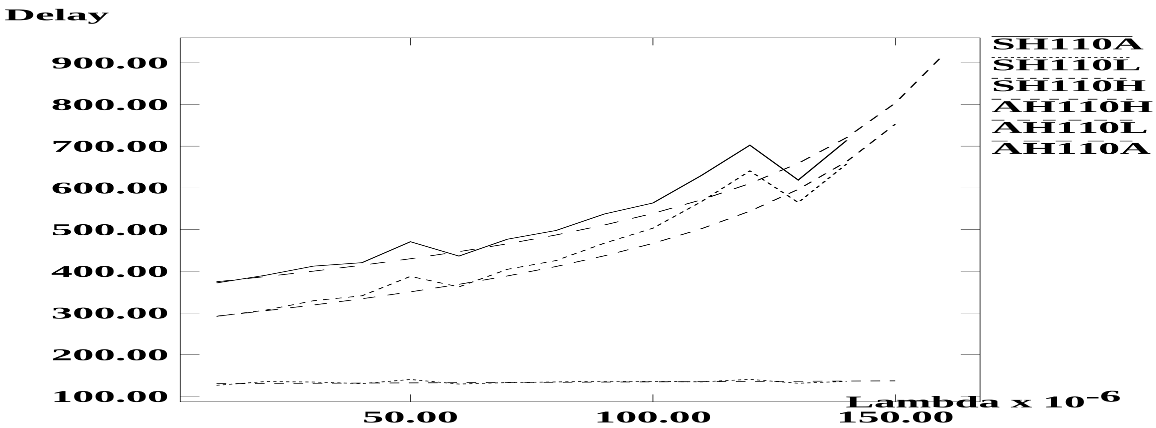
Figure 7.3: Simulation and analysis of hop, loop and avg delays in HTL, LTH and NOS for MCDL h=7



(a). Hop. loop and avg delays for h=2



(a). Hop, loop and avg delays for h=5



(a). Hop, loop and avg delays for h=10

Figure 7.4: Simulation and analysis in exponentially distributed packet length for MCDL h=2,5 and 10 in HTL

Chapter 8

FAULT TOLERANCE STUDIES

The problem of routing when a node or a link being faulty is considered here. In Chapter 3, the number of long and short hops in a shortest path between nodes were found. It was also said in Section 2.2 that the order in which these links are used to reach the destination is immaterial. But in case of node or link failure, will all the possible orderings of the links take to the destination avoiding faulty node/edge? In order to answer this, two cases of knowledge on fault information for both the node and link are considered here. They are: *global* knowledge and *neighbourhood* knowledge of faults. For simplicity let nodes 0, i, *and* j ($\neq\ i$) represent the source, destination and faulty nodes respectively and the shortest path to i *and* j from 0 lie in the $(+h, +1)$ link combination. The discussions given in Sections 8.1 and 8.2 hold even if the shortest paths lie in the other three link combinations. In the case of global knowledge of faults, the address of faulty node, j is communicated to all other nodes in the network. In the case of neighbourhood knowledge of faults, the location of faulty node is known only when a node tries to send a packet to it. Similarly the knowledge on faulty edges are also considered.

8.1 Routing under node faults

8.1.1 Global knowledge of node faults

Let $w_i[+h] + y_i[+1]$ be the shortest path to i. Let the symbol 0 represent the traversal along a $[+1]$ link and the symbol 1 the hop traversal along the $[+h]$ link. A shortest path is denoted as a sequence of w_i 1's and y_i 0's in any order. Thus the number of such orderings are $\binom{w_i + y_i}{y_i}$ or $\binom{w_i + y_i}{w_i}$. Each of these permutations is a shortest path. We represent the paths to i in a two dimensional plane as shown in Figure 8.1. A traversal in the grid along the horizontal and vertical lines represent the loop and hop traversals respectively. The four link combinations *viz*, $(+h, +1)$, $(+h, -1)$, $(-h, -1)$ and $(-h, +1)$ form the four quadrants in the two dimensional plane. A grid with $w_i = 9$, $y_i = 10$ and a path to i areshown in Figure 8.1.

If the shortest path to node j lies in the other three link combinations, then none of these permutations find the faulty node in the paths to i. Thus it is guaranteed that destination i can be reached without passing through the faulty node j. But, let the shortest path to j also be in $(+h, +1)$ link combination and is $w_j[+h] + y_j[+1]$. Again, if $w_j > w_i$ and $y_j > y_i$ then also none of the the permutations will find j in the path to i. We omit the case of $w_j = w_i$ and $y_j = y_i$ as we assumed $j \neq i$.

If $w_j \leq w_i$ or $y_j \leq y_i$ (refer figure 8.1), then the faulty node can be in one of the shortest paths. Hence to avoid the faulty node, those permutations which have the first $(w_j + y_j)$ length of the sequence containing any permutation that is formed with w_j 1's and y_j 0's is to be discarded as shown in figure 8.1. The approach given in Algorithm 8.1 avoids the faulty node.

ALGORITHM 8.1 *// To obtain a sequence of links that avoids the faulty node whose address is known a priori globally: Source is 0, destination is i and faulty node is j.//*

1. *Either $w_j = 0$ or $y_j = 0$.*

   (a) *If $w_j = 0$ and $w_i > 0$, the link sequence is:*
   a 1 followed by a permutation of $(w_i - 1)$ 1's and y_i 0's (refer case 1(a) in figure 8.2).
   END of link-sequence generation. $\bigtriangledown$

   (b) *If $y_j = 0$ and $y_i > 0$, the link sequence is:*
   a 0 followed by permutation of w_i 1's and $(y_i - 1)$ 0's (refer case 1(b) in figure 8.2).
   END of link-sequence generation. $\bigtriangledown$

2. *If $0 < w_j \leq w_i$ and $0 < y_j \leq y_i$, the link sequence is:*
   any permutation of $(w_j - 1)$ 1's and $(y_j - 1)$ 0's (to be followed by (a) or (b) in the following)

   (a) *Either $y_i - y_j > 1$ or $y_i - y_j = 1$.*

      i. *If $y_i - y_j > 1$, continuing the link-sequence:*
      two 0's followed by any permutation of $(w_i - w_j + 1)$ 1's and $(y_i - y_j - 1)$ 0's (refer case 2(a)-i in figure 8.2).
      END of link-sequence generation. $\bigtriangledown$

      ii. *If $y_i - y_j = 1$, continuing the link-sequence:*
      two 1's followed by any permutation of $(w_i - w_j - 1)$ 1's and a 0 (refer case

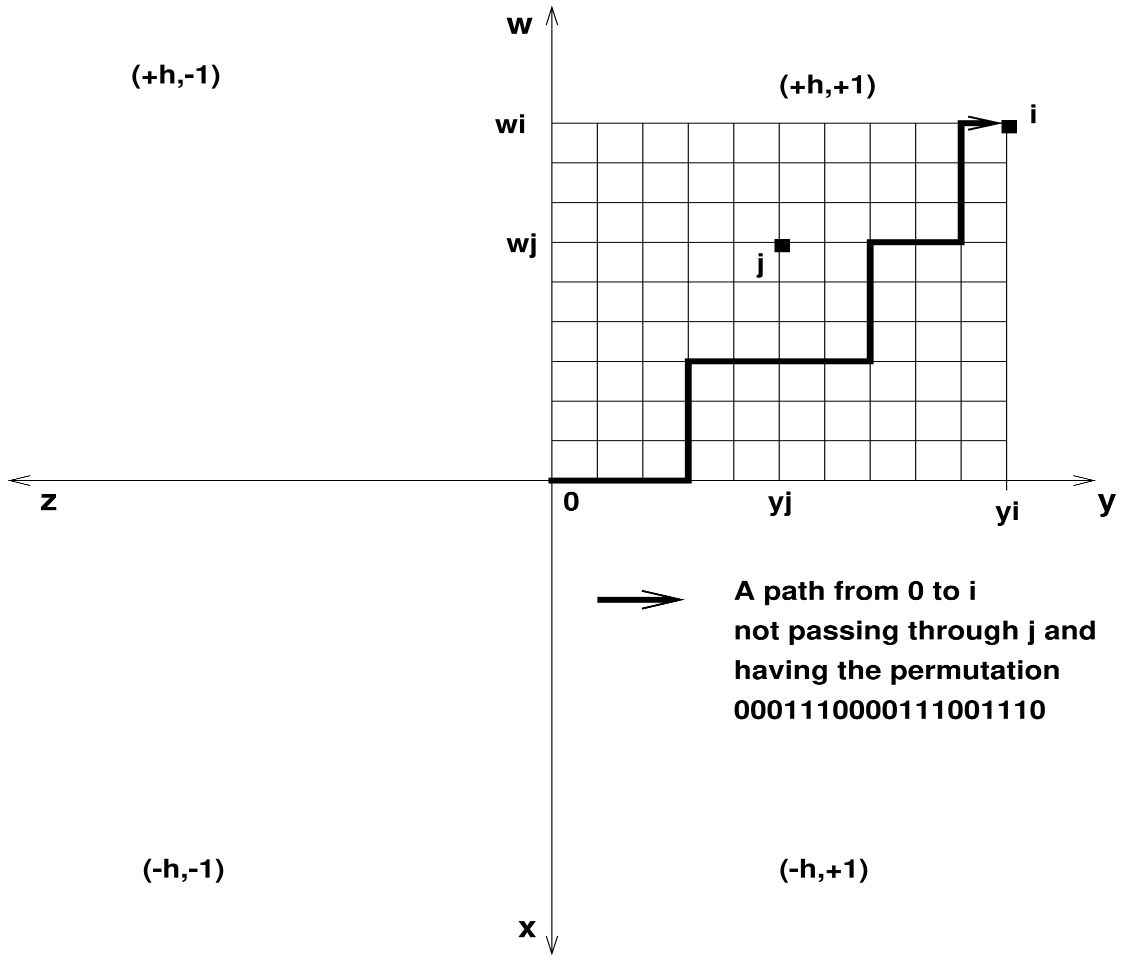


Figure 8.1: A shortest path avoiding faulty node

2(a)-ii in figure 8.2).
END of link-sequence generation. $\bigtriangledown$

(b) *Either* $w_i - w_j > 1$ *or* $w_i - w_j = 1$

i. *If* $w_i - w_j > 1$, *continuing the link-sequence:*
two 1's followed by any permutation of $(y_i - y_j + 1)$ *0's and* $(w_i - w_j - 1)$ *1's (refer case 2(b)-i in figure 8.2).*
END of link-sequence generation. $\bigtriangledown$

ii. *If* $w_i - w_j = 1$, *continuing the link-sequence:*
two 0's followed by $(y_i - y_j - 1)$ *0's and a 1 (refer case 2(b)-ii in figure 8.2).*
END of link-sequence generation. $\bigtriangledown$

3. *If* $w_j = w_i = 0$ *and* $y_j < y_i$, *the link-sequence is:*

(a) $(y_j - 1)\ 0's,\ 1\ 0\ 0,\ a[-h] traversaland (y_i - y_j - 1)\ 0's$, *which means:*
Travel along the $[+1]$ *links up to the node* $(j-1)$, *take* $1[+h], 2[+1], 1[-h]$ *and then continue to use the* $[+1]$ *links for the remaining distance (if any) up to* i. *Note that this route increases shortest path by 2 (refer case 3(a) in figure 8.2).*
END of link-sequence generation. $\bigtriangledown$

(b) *If* $y_j = y_i = 0$ *and* $w_j < w_i$, *the link sequence is:*
$(w_j - 1)\ 1's,\ 0\ 1\ 1,\ a[-1] traversaland (w_i - w_j - 1)\ 1's$, *which means:*
Travel along the $[+h]$ *links up to the node* $(j-1)$, *take* $1[+1], 2[+h], 1[-1]$ *and then continue to use the* $[+h]$ *links for the remaining distance, if any (refer case 3(b) in figure 8.2).*
END of link-sequence generation. $\spadesuit$

The link-sequence can be placed in the header of a packet along with the path length $(w_i + y_i)$. When a packet reaches any intermediate node, the appropriate link is selected from the link-sequence and the path length is decremented. The packet is copied at a node where path length value becomes 0.

This global knowledge places a communication overhead in the network as some broadcast mechanism is to be implemented to communicate the faulty node locations.

8.1.2 Neighbourhood knowledge of node faults

In this case while data is in transit, a node en route knows whether its adjacent node in the shortest path is faulty or not. The total number of links (both hop and loop) are precomputed at the source. These link numbers may be placed in the header of the message at the source and the routing is done as follows. In the following it is assumed that node 0 is source, node i is destination and its shortest path lies in $(+h, +1)$ link combination with a path of $w_i[+h] + y_i[+1]$. The other three link combinations can be argued similarly.

As the message reaches a node d, it chooses its next out going link according to shortest path algorithm, decrements the corresponding number of hop or loop links by one and forwards the message on that selected link. This process is repeated until either the destination is reached or a faulty node is encountered. Now, let at some intermediate node d, the node adjacent to it is faulty. Note that at this node, the remaining number of hop and loop links to be covered (w_{i-d} and y_{i-d}) are in the header of the message. We give an algorithm to select the out going link. This case is called neighbourhood knowledge on faults.

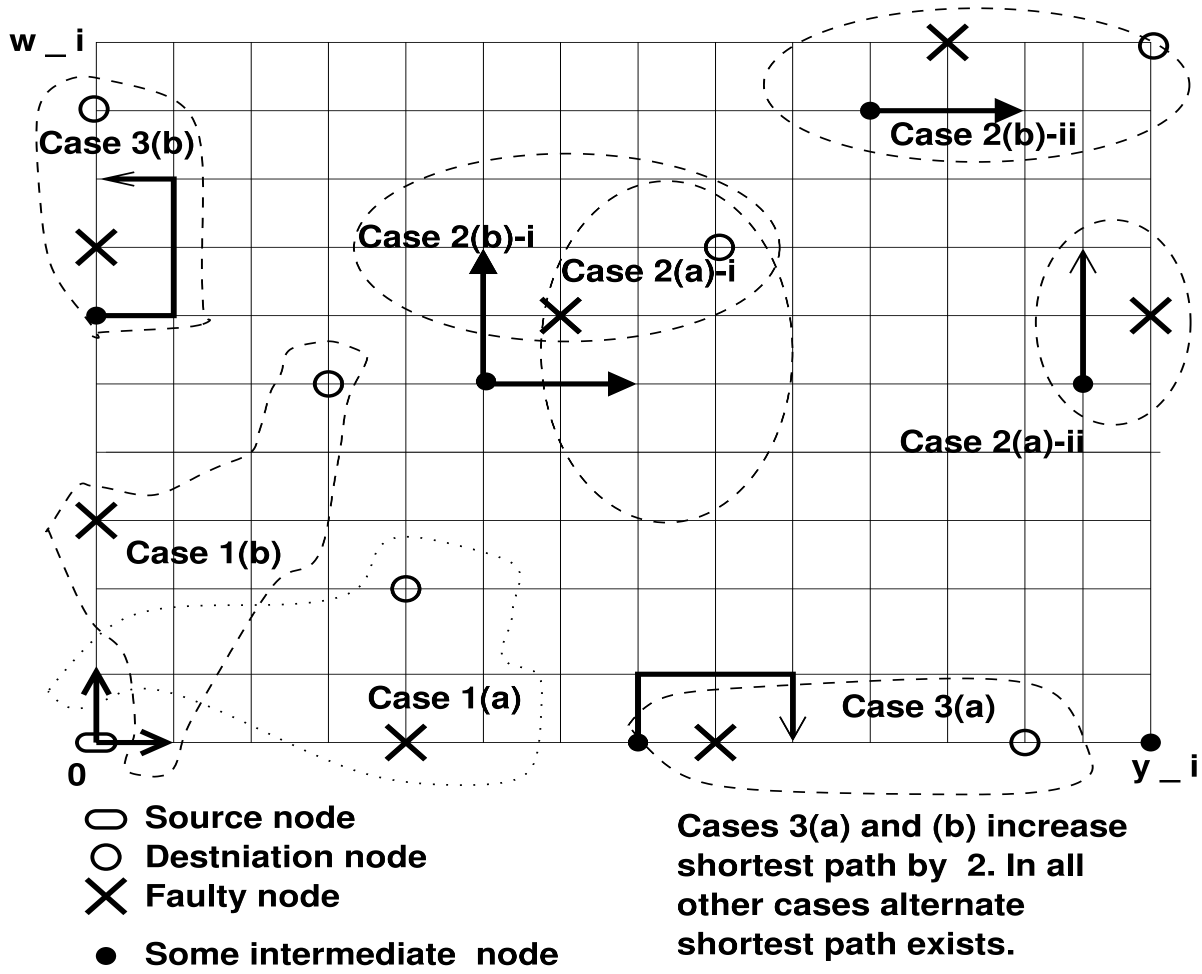


Figure 8.2: Alternate shortest paths in node failures (for Algorithm 8.1)

ALGORITHM 8.2 *// To find the choice of links en route that avoids the faulty node whose address is not known a priori//*

1. *Let node* $(d+h)$ *is faulty.*
   If $y_{i-d} \neq 0$
   The [+1] link is chosen to avoid the faulty node. ▽
   If $y_{i-d} = 0$,
   Since the path up to d is covered, for the remaining distance of (i − d) do the following: choose a path segment: $[+1]$, $[+h]$, $[+h]$, *and* $1[-1]$. *This avoids the faulty node* $(d+h)$ *and leads to node* $(d+2h)$. *However this segment increases the shortest path length by 2.* ▽

2. *Let node* $(d+1)$ *is faulty.*

Similar to the argument given in the above choose a [+h] link (if $w_{i-d} \neq 0$) or follow a path segment $[+h],\ [+1],\ [+1],\ [-h]$ with if $w_{i-d} = 0$.

Similarly the cases of faulty $(d - h)$ and $(d - 1)$ nodes can be explained. ♠

The disadvantage of the neighbourhood knowledge is when an adjacent node is faulty and only one link type (either hop or loop) is available to reach the destination by the shortest path, then the path length increases. Where as in such cases, with the global knowledge, an alternate shortest path may have been possible since the path and the sequence of links to be used are computed at the source before transmission. Hence neighbourhood knowledge may increase the path length even though the faulty node could have been avoided with a priori information as in global knowledge of the location of the faulty node.

8.2 Routing under edge faults

In $G(N; 1, h)$ the four outgoing links of node j can be distinguished as: $j_{+1},\ j_{-1},\ j_{+h},\ j_{-h}$. When routing messages under shortest path in $(+h, +1)$ link combination (the other three link combinations can be similarly explained), the faulty links are bypassed like in node faults. The link faults may either be known globally (in the case of $global\ knowledge\ of\ faults$) or be known only to the neighbouring nodes ($neighbourhood\ knowledge\ of\ faults$). In the former case the link sequence can be computed at the source and accordingly the message can be routed bypassing the faulty link. In the latter case they are by passed when encountered en route. We use the notation that node 0 is source, node i is destination and the faulty link could be either of the outgoing four links connected to node a node j, *i.e.*, $j_{+h},\ j_{-h},\ j_{+1},\ j_{-1}$.

8.2.1 Global knowledge of edge faults

Like in Section 8.1.1, for the global knowledge of edge faults some broadcast mechanism is assumed to be present to communicate the faulty edge addresses to all the nodes in the network. The following methodology finds a routing under global knowledge of edge faults. Like in the Algorithms 8.1 and 8.2, $w_i[+h] + y_i[+1]$ and $w_j[+h] + y_j[+1]$ are the shortest paths to nodes i and j respectively.

ALGORITHM 8.3 *// To obtain a sequence of links that avoids the faulty edge whose address is known a priori//*

1. *In the following cases corresponding faulty link will not appear in the shortest path to i.*

	Condition	Type of link fault
(a).	$w_j > w_i$ or $y_j > y_i$ or $j = i$	Any of the links
(b).	$w_j = w_i$	j_{+h}
(c).	$y_j = y_i$	j_{+1}

2. *In the following cases the corresponding faulty link is likely to appear in the shortest path. But this can be bypassed using an alternative shortest path.*

	Condition	Type of link fault
(a).	$0 \leq w_j < w_i$ and $0 \leq y_j < y_i$	Any of the link
(b).	$w_j = w_i$, $0 \leq y_j < y_i$	j_{+1}
(c).	$y_j = y_i$, $0 \leq w_j < w_i$	j_{+h}

3. *In the following cases the alternative path under the fault increases the shortest path.*

	Condition	Type of fault
(a).	$w_i = 0$, $0 \leq y_j < y_i$	j_{+1}
(b).	$y_i = 0$, $0 \leq w_j < w_i$	j_{+h}

In (a). above, one alternative path at node j is $1[+h]$, $1[+1]$, $1[-h]$ which increases shortest path by 2.
Another alternative is to find shortest path sp_{--}, sp_{+-} in $(-h, -1)$ and $(+h, -1)$ link combinations and $Min\ \{sp_{--},\ sp_{+-}\}$. This may give a path that is in only one excess than the shortest path length.
Similarly case (b), above can be explained.

With the above three cases the suitable link sequence can be generated. ♠

8.2.2 Neghbourhood knowledge of edge faults

Let the shortest path to i from 0 be in $(+h, +1)$ link combination. So a node has the choice of selecting $[+h]$ or $[+1]$ links. Assume that one of them is faulty and its address is known to only the node from which it is directed. With this knowledge of faults, the next outgoing link at a node is chosen arbitrarily. When the selected link is faulty, the other alternative link is chosen in accordance with a routing algorithm. The disadvantage of neighbourhood knowledge on faults is that it increases path length even if an alternate path were available under global knowledge.

Chapter 9

WORMHOLE ROUTING IN MCDL

We study MCDL for deadlock-free wormhole routing in direct network applications, using the theory developed in in [DS87]. In [BC96, DA93], a routing function was defined for a virtual interconnection network and then a channel dependency graph was derived from this routing by using it as an implicit definition of all possible connections [DS87]. Then it was proved that if the channel dependency graph is acyclic, the routing function was said to be deadlock-free. In contrast, in [LH91] the physical interconnection network is used to define the possible cyclic channel dependencies and *virtual channels* are introduced to avoid them. Now any routing function is restricted to use these channels of virtual interconnection that are acyclic, so that the routing is deadlock-free. But in our work we keep the construction of channel dependencies simple by hierarchical decomposition of the physical interconnection. In addition if a routing function adds new dependencies, it may still be possible to argue the deadlock freedom. Thus, instead of showing each routing function deadlock-free, the virtual interconnection network is shown to be deadlock-free for a given channel dependency graph and any routing function that obeys these restrictions of this channel also deadlock-free.

With this background, we consider *directed array, directed ring* and *undirected ring*. We and develop deadlock-free routings with virtual channels in them. We then extend these deadlock-free routing results for *MCDL* topology with virtual channels.

9.1 MCDL as direct network topology

Several studies on *k-ary n-cube, mesh, tori* and *hypercube* were done in direct network topology as interconnection networks for multicomputer and multiprocessors [DA93, DS87, LH91, DYN97]. In the following we see how best the MCDL topology suits the direct network topological characteristics.

The MCDL is a fixed degree (four) network with hop and loop connections at every node and packets travel one or more intermediate nodes before reaching their destination nodes. A node has four channels that are bidirectional. The routing is of *source routing* in which the source node determines the entire path before sending the packet (refer the shortest path routing explained in Section 3.3 and 3.4). Each packet can carry this information along with it to select either a hop channel or a loop channel at a node. The routing can be *distributed* also in the sense that while forwarding the received packet from the input channel, the next outgoing output channel can be determined by invoking the shortest path routing algorithm. The routing guarantees delivery of packets at the destination. The hop and loop buffers can be allocated to the respective incoming channels without any buffer contention in both the the *hop to loop (HTL)* and *loop to hop (LTH)* routings as explained in Section-5.2.

The routing algorithms in direct networks generally determine which output channel is selected for a packet arriving on a given input channel. This can be referred to as *output selection* policy. Similarly, an outgoing channel can be requested by packets arriving from many different input channels, and is referred to as *input selection policy*.

In MCDL, the output selection policy is achieved through the HTL and LTH routings. In these routings, one of the output channels at a node can be requested by packets received on an input channel based on the routings. The input selection policy is achieved through the shortest path. In the shortest path, depending on the link combination used by a packet an outgoing link can be selected by any of the input channels of a node. Moreover, an output channel that was requested by several messages is assigned using first-cum-first served policy (refer the $h-$buffer and $1-$buffer managements in Chapter 5).

9.2 Wormhole switching in MCDL

We explained in Section 2.6.2 that wormhole switching has network latency independent of distance between nodes. Also another advantage of wormhole switching is that it requires small buffer sizes at nodes for the blocked packets. We see below wormhole switching technique and its network latency.

In wormhole switching every packet is divided into a number of flits (flow control digits). The header flit determines the route. As the header advances along the specified route, the remaining flits follow in a pipelined fashion. Hence if a communication channel transmits the first flit of a message, it must transmit all the remaining flits of the same message before transmitting flits of another packet. Different packets can be interleaved during transmission.

However, flits from different packets can not be mixed up. The main advantages of wormhole switching are low memory requirements in routers, pipelined data movement and low communication latency. Let L and F be the length of packet and flit (in bits). Let W be the channel bandwidth in bits/sec. If the packet has to travel a distance of D links , the the latency is given by $\frac{L}{W} + \frac{F}{W} \cdot D$. If the size of flit is such that $F \ll L$ then the latency is independent of D the distance between the source and destination.

The main disadvantage of wormhole switching is channel congestion, since a blocked message does not relinquish the communication channels it has acquired. Hence one of the important issues in the design of routing algorithm is to reduce the channel congestion. This can be achieved by the use of virtual channels introduced in Section 9.3. We see in Section 9.4 that virtual channels avoid the occurrence of *deadlocks*.

9.3 Virtual channels

In the wormhole switching, we saw that messages or parts of messages were buffered at the input and output of each physical channel. Therefore once a message occupies a buffer for a channel, no other message can access the physical channel. Alternatively, if the physical channel can support several *logical channels* or *virtual channels* the blocking can be reduced. These virtual channels can be time multiplexed over the single physical channel. In Figure 9.1 each unidirectional physical channel multiplexes two virtual channels. Consider a wormhole message packet in each virtual channel. Each message can share the physical channel on flit-by-flit basis. The physical channel protocol should be able to distinguish between the virtual channels using the physical channel. Logically each virtual channel operates as if each were using a distinct physical channel. The conceptual model for the communication portion of a node is a set of queues: one for each input channel. Instead of one channel dumping a message into a buffer and that buffer being emptied into another channel, the queue of an input channel is connected to an output channel, and the message flits flow through the connection until the entire message is passed. Thus the connection can be broken and other other connections can be made.

Virtual channels were introduced in the study of deadlocks in wormhole switching as to be explained in Section 9.4.1. Virtual channels can also be used to improve message latency. By allowing messages to share a physical channel, they can make progress rather than remain blocked. Thus the overall time a message spends blocked at a router waiting for a free channel is reduced.

Figure 9.2 illustrates the concept of virtual communications under a single physical communication system. The physical channels are represented with solid lines and virtual channels

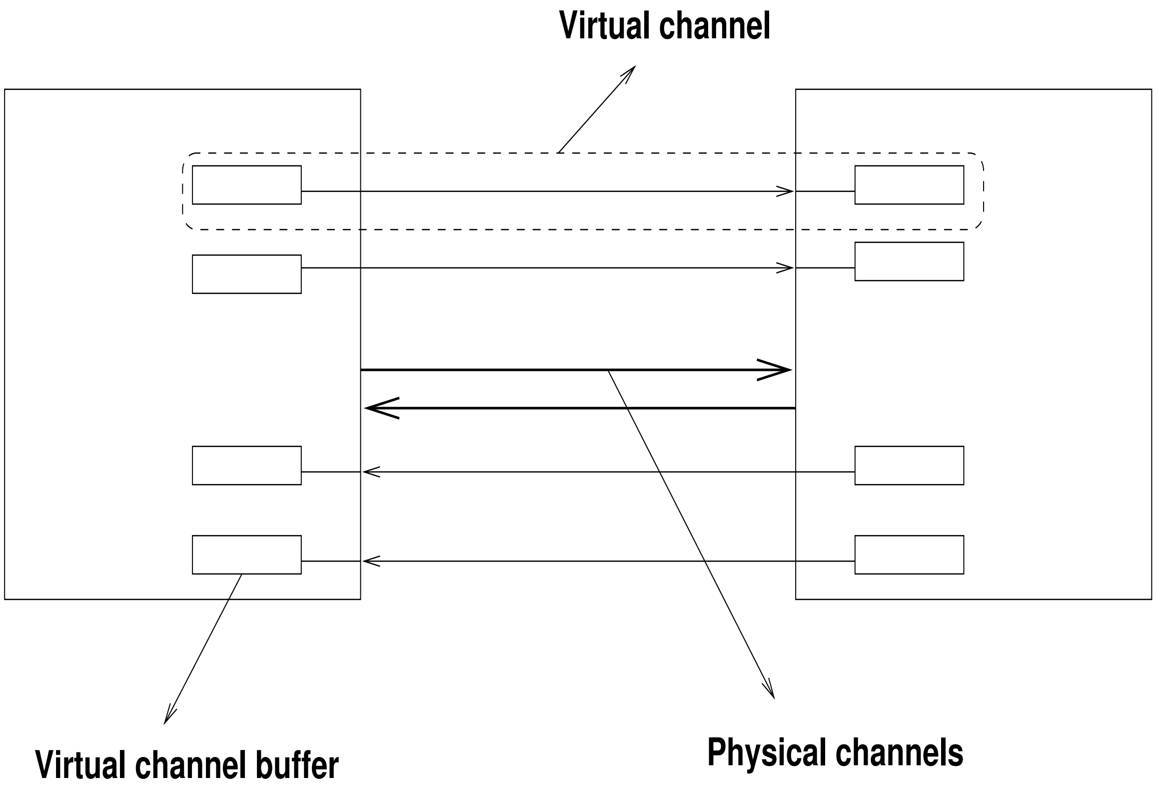


Figure 9.1: Virtual channels

with dotted lines in Figure 9.2. There is a mapping from the virtual nodes and channels to the physical nodes and channels. The obvious restriction in the mapping is that nodes adjacent in the physical system are mapped to the nodes adjacent in the virtual system. Two virtual channels are mapped on the physical channels $(1, 2)$ and $(3, 4)$. Only one virtual channel is mapped for the edges $(2, 3)$, $(1, 4)$ and $(4, 5)$. All virtual channels that map to a physical channel are multiplexed over the physical channel. By grouping some subset of the total number of virtual channels we can configure different *virtual interconnections*. For example in Figure 9.2 the virtual channels $(1, 2)$, $(2, 3)$ and $(3, 4)$ (in dashed lines) form a virtual interconnection. Similarly the virtual channels $(1, 2), (1, 4).(3, 4)$ and $(4, 5)$ (in dotted lines) form another virtual interconnection. We give below some definitions needed for our work, based closely on the work in [DS87] Increasing number of virtual channels has a direct impact on the router performance [DYN97]. The router now becomes more complex since it must support arbitration between multiple virtual channels. The number of inputs and outputs that must be switched at each node is increased, thus increasing the switch (or router) complexity.

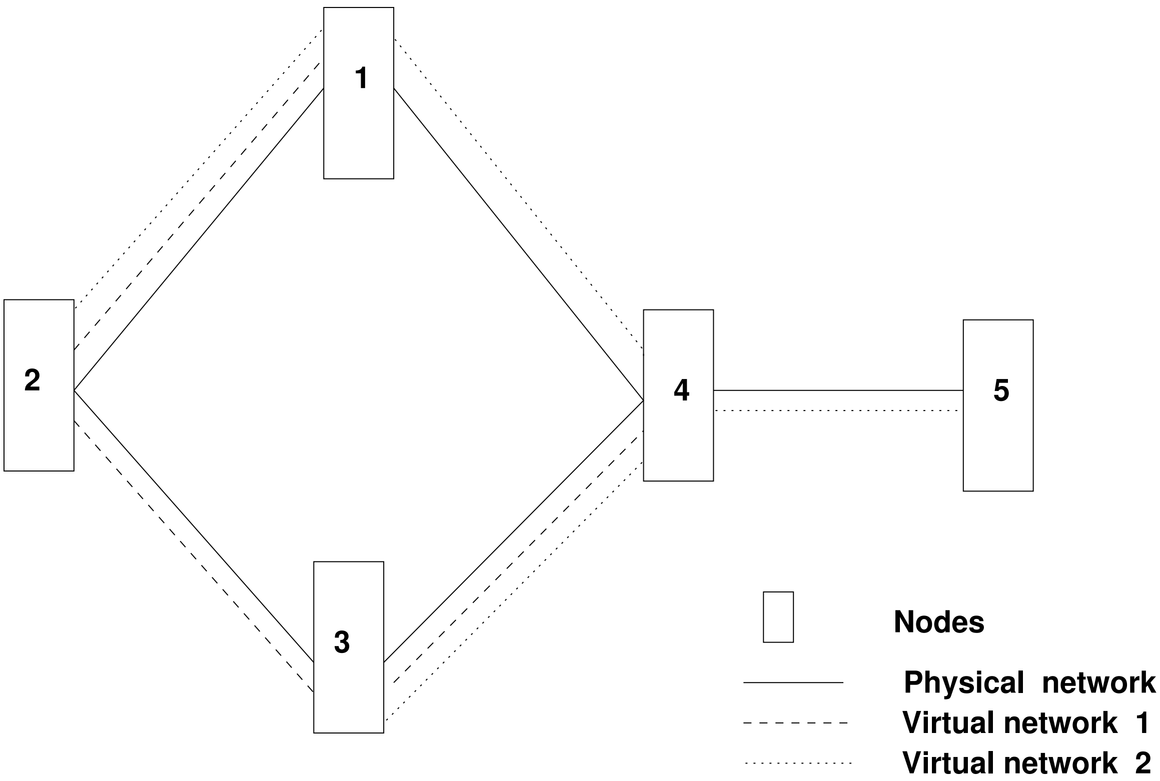


Figure 9.2: Virtual interconnections in a physical network

9.4 Deadlocks in MCDL

Switching strategy and routing algorithms are among several factors that affect communication latency. One situation that can postpone packet delivery indefinitely is *deadlock*. The possibility of deadlock is introduced when a message must travel through intermediate nodes because some portion of the message must always be stored at these nodes. Since storage is finite, it may become filled. When this happens, new messages can not enter the node until old messages start to leave. Intuitively, deadlock occurs when messages traveling in the communication system develop *dependency loops* among themselves that prevent further movement.

In MCDL there are g number of undirected hop rings and an undirected loop ring. In these $(2g+2)$ directed rings, movement of flits cause deadlock. For example consider a directed hop ring. A flit traveling in one $[+h]$ link from a node say i, requires the next link from the outgoing links of node $(i+h)$. Let a $[+h]$ link from this node $(i+h)$ is the next link that this flit needs. We note this as channel dependency. If such dependencies exist among all the consecutive hop links of that hop ring, then messages will enter into deadlock and they will never be delivered to their destinations. Thus we need to study such dependencies among all the links of the $(2g+2)$ rings in MCDL and design a deadlock-free routing scheme.

9.4.1 Handling of deadlocks

There are three strategies for handling deadlocks. They are: *deadlock prevention, deadlock avoidance* and *deadlock recovery.* In deadlock prevention, channels are granted to a packet in such a way that a request never leads to a deadlock. It can be achieved by reserving all the required channels before starting a packet transmission. In deadlock avoidance, channels are requested as packet advances through the network. However, a channel is granted to a packet only if the resulting global state is safe. This strategy requires nodes to send additional packets to update the global state. Achieving this in a distributed manner is not an easy task. In deadlock recovery strategy, channels are granted without any check. Therefore some deadlock is possible and detection mechanism must be provided. If deadlock is detected, some channels are deallocated and granted to other packets. In order to deallocate channels, packets holding in those channel buffers are usually aborted. Deadlock prevention method is conservative. Reserving the channels before commencement of transmission may lead to low channel utilization. Deadlock avoidance strategies are less conservative because channels are requested only when they are really needed. Finally, deadlock recovery strategy is optimistic. They can only be used when deadlocks are rare to occur. Here we study deadlock a deadlock avoidance method for MCDL.

Deadlock avoidance can be achieved by restricting the combination in which the input and output channels are connected. This combination of input/output channel selection can be facilitated with the use of *virtual channels* between pairs of nodes. Multiple virtual channels between a pair of nodes is provided by multiplexing the available bandwidth of the single physical channel as discussed in Section 9.3. The routing algorithm specifies a relation R on virtual channels, where R determines which path and channels are suitable for deadlock-free routing for the next hop of the message. From now on we use routing relation and routing algorithm synonymously. We use *channel dependency graph* of the routing relation R. The channel dependency graph, henceforth dependency graph, of R is obtained as follows. The vertices of dependency graph are the virtual channels, there is a directed edge from vertex r_1 to r_2 if a message can use r_2 immediately after using r_1. For a deadlock-free operation, the channel dependency graph should be acyclic. We consider only minimal routing of messages so that the bandwidth used per message is minimum. We avoid starvation by assigning channels on FCFS basis.

9.5 Deadlock theory

DEFINITION 9.1 *A physical interconnection network PI, is a strongly connected directed graph $PG(PN, PC)$. PN is a set of nodes representing processors and PC is a set of edges repre-*

senting actual physical channels connecting the processors. ♠

Let pn_i and pc_i be the i^{th} physical node and channel respectively.

DEFINITION 9.2 *A virtual interconnection network VI_j is a directed graph $VG_j(VN_j, VC_j)$. VN_j is a set of virtual nodes and VC_j is set of virtual channels.* ♠

The subscript j is used because a single physical interconnection could have several virtual interconnections. VN_j is a set of virtual nodes that are mapped to PN by the function $nmap: \ VN_j \rightarrow PN$. VC_j is set of edges representing virtual channels that are mapped to PC by the function $cmap: \ VC_j \rightarrow PC$. Let vc_{ij} be the i^{th} virtual channel in the j^{th} virtual interconnection.

DEFINITION 9.3 *A connection can exist between vc_{ij} to vc_{kj} if the flit traveling on vc_{ij} can immediately use vc_{kj} after traveling on vc_{ij}.* ♠

DEFINITION 9.4 *A channel dependency graph D_j for the virtual interconnection network VI_j is a graph $D_j = G_j(VC_j, E_j)$ where $E_j = \{(vc_{ij}, vc_{kj})|$ a connection can exist from vc_{ij} to vc_{kj} $\}$.* ♠

Thus D_j is a graph with a node for every channel in VI_j and edges to define what connections can be made between the channels. A channel is not allowed to connect to itself, so there will be no $1-$cycles in D_j.

DEFINITION 9.5 *A routing function $R_j: \ VC_j \ \rightarrow VC_j$ for a virtual interconnection network VI_j maps a channel vc_{ij} in the state of VI_j to another channel vc_{kj}.* ♠

When the first flit of a message arrives from the channel vc_{ij}, the routing function determines the channel vc_{kj} that vc_{ij} should be connected to so that the flits can advance (possibly if the arriving flit is not blocked due to the fact that vc_{kj} is busy) toward their destination. If the next immediate channel vc_{kj} is busy, then the flit arriving from vc_{ij} is buffered at the queue available for vc_{kj}. If a connection is possible from vc_{ij} to vc_{kj} then the motion of the flits in vc_{ij} may be dependent on the motion of the flits in vc_{kj}. This dependency can be cyclic among two or more channels and deadlocks occur. Deadlocks result when flits wait for other flits to move. If one flit is forced to wait on another to move, its motion is dependent on the motion of the other flits. That is why D_j which specifies channel connections is called a channel dependency graph. It is important to note that once D_j is proven to be acyclic, then there is no deadlock within VI_j. Deadlock is impossible in PI if the overall channel dependency graph is acyclic [DS87]. By introducing virtual channels the cyclic dependency can be avoided resulting deadlock-free routing.

9.5.1 Deadlock avoidance

In [DS87], a routing function was defined for a virtual interconnection network and then a channel dependency graph was derived from this routing by using it as an implicit definition of all possible connections. The channel dependency graph is shown to be acyclic and thus the routing function is deadlock free. In contrast, we take [LH91] the channel dependency graph as the definition of what connections are possible in the among physical channels in the physical interconnection. The physical channel connections which have cyclic dependencies are provided with virtual channels and a restriction on using those virtual channels is given. Now the interconnection with virtual channels (called *virtual interconnection*) is shown to be acyclic. Hence any routing function that uses this virtual interconnection with the specified restriction in using the virtual channels is deadlock free. Thus, instead of showing each new routing function deadlock-free, the virtual interconnection network is shown to be deadlock-free for a given channel dependency graph and the new routing function are shown to obey the restrictions of the channel dependency graph. We use the following result also given in [Car79] to prove that a graph is acyclic.

DEFINITION 9.6 *If a number $\theta(A)$ can assigned to each node A in a graph such that an edge from A to B implies $\theta(A) < \theta(B)$, then the graph is acyclic.* ♠

This result is used to show that the following networks are deadlock-free.

1. Directed linear array
2. Directed ring virtual channels.
3. Undirected ring with virtual channels.
4. Undirected MCDL with virtual channels.

9.6 Directed linear array

The physical interconnection network directed array $DA(PN, PC)$ is shown in Figure 9.3. The physical nodes are identified by a label pn_i, $pn_i \in PN$. The physical channels by pc_i, $pc_i \in PC$, where i is the node position from where flits enter the channel. The node position i is labelled $0, 1, \ldots (n-1)$ for an n node DA.

THEOREM 9.1 *The channel dependency in DA is acyclic.*

Proof: In the physical interconnection shown in Figure 9.3, a flit arriving from channel pc_i will acquire pc_{i+1} next, $0 \leq i < n-1$. So the channel dependency graph is $D(PC, E)$, where $E = \{(pc_i, pc_{i+1})\}$ $0 \leq i < n-1$. Such a dependency among the channels is numbered as

$$\theta(pc_i) \quad = \quad k_i,$$

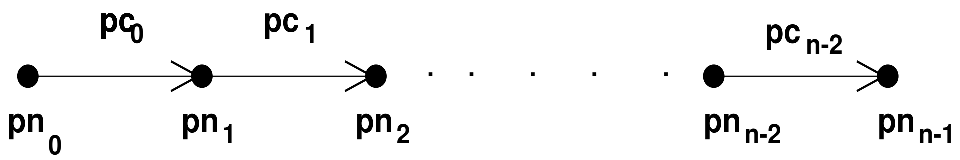


Figure 9.3: Physical interconnection in a directed array

where θ is an integer valued function such that $\theta(pc_i) < \theta(pc_{i+1})$ whenever pc_i depends on pc_{i+1}. Thus $k_i < k_{i+1},\ 0 \leq i < n-1$. So there exist k_i's such that $k_0 < k_1 < \ldots < k_{i-2}$. Since we can number all the physical channels in this manner, as per the Definition 9.6, the channel dependency D of DA acyclic. ♠

Let a routing function be $R_{DA} : PC \rightarrow PC$ such that $(pc_i, pc_{i+1}) \in R_{DA},\ 0 \leq i < n-1$. Since R_{DA} uses the channel of D, R_{DA} is deadlock-free.

9.7 Directed linear array with wraparound

This interconnection is also a directed ring $DR(PN, PC)$ and is shown in Figure 9.4, where $pn_i \in PN$ are the physical nodes and the i's are labelled $0, 1, \ldots (n-1)$ for an n node DR. The $pc_i \in PC$ represent the physical channel where i is the node position from where flits enter the channel. In the following theorem we prove cyclic dependency among the channels of DR and the existence of a deadlock among them.

THEOREM 9.2 *The channel dependency graph in the physical interconnection of DR is not acyclic.*

Proof: A flit arriving from channel pc_i will acquire $pc_{\overline{i+1}}$ next, where $\overline{i+1} = (i+1)$ mod n and $0 \leq i \leq n-1$. Thus there exist a channel dependency from pc_i to $pc_{\overline{i+1}}$. So the channel dependency graph is $D(PC, E)$, where $E = \{(pc_i, pc_{\overline{i+1}})\},\ 0 \leq i \leq n-1$. Such a dependency among the channels is numbered as

$$\theta(pc_i) = k_i$$

where θ is an integer valued function such that $\theta(pc_i) < \theta(pc_{\overline{i+1}})$ whenever pc_i depends on $pc_{\overline{i+1}}$. Thus $k_i < k_{\overline{i+1}},\ 0 \leq i \leq n-1$. So there do not exist k_i's such that $k_0 < k_1 < \ldots < k_{n-2} < k_{n-1} < k_0$. Since we can not number all the physical channels, as per the Definition 9.6, the channel dependency D of DR is not acyclic. ♠

Since the dependency graph is cyclic, DR is not deadlock-free. The reason for cyclic channel dependency (and consequently a deadlock) is due to the channel from pn_{n-1} to pn_0.

DEFINITION 9.7 *The channel from pn_{n-1} to pn_0 is the wraparound channel in DR.* ♠

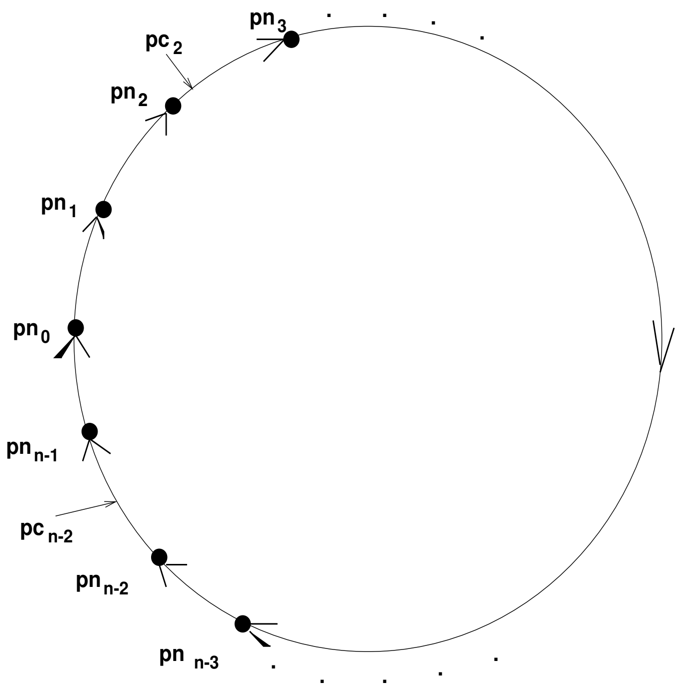


Figure 9.4: Directed ring physical interconnection

Let y be the distance from pn_i to pn_j, *i.e.*, $y = (j - i)\ mod\ n$. Then if $(i + Y)\ mod\ n = 0$ for some $Y \leq y$, we say that the path will contain the wraparound channel. In other words if i and j are the source and destination addresses of a packet and $i > j$, then we say that the path contains the wraparound channel. Now we describe a virtual interconnection network $DRVI$ for the directed ring DR to avoid cyclic channel dependencies.

The precise description of $DRVI$ has two components: Identifying virtual nodes and virtual channel that constitute a virtual topology and the other is listing of the edges for the channel dependency graph that restricts the routing functions under consideration. We introduce two virtual channels for every physical channel pc_i of DR in $DRVI$. Call one of them as *level 1* virtual channel and the other *level 0* virtual channel. If a flit from a node i is destined to another node j $(j > i)$, then we allow it to use level 0 virtual channel. But, if $j < i$, then we allow it to use level 1 virtual channels. Thus we separated the channel dependencies among the channels based on their destination addresses. We see in the following, the definition of DRVI and later show that this does not produce any cyclic dependencies.

The virtual interconnection $DRVI(VN, VC)$ where VN is set of virtual nodes and VC is the set of virtual channels is shown in Figure 9.5. While describing the virtual nodes and channels we add a new subscript to them L, which is the level number. Define the mappings $nmap : VN \rightarrow PN$ and $cmap : VC \rightarrow PC$. All virtual nodes of DRVI are mapped as: $nmap(vn_{i,L}) = pn_i$. Similarly, all virtual channels are mapped as: $cmap(vc_{i,L}) = pc_i$ except

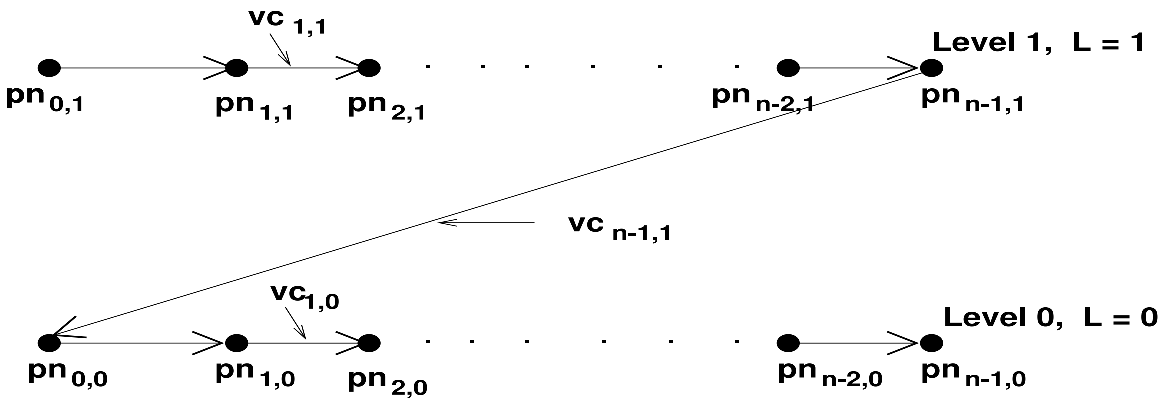


Figure 9.5: Virtual interconnections in a directed ring

in the case of $i = n-1,\ L = 0$ in which case the *cmap* is undefined (refer Figure 9.5). We represented in the figure all the level 1 nodes and channels in the upper line and level 0 nodes and channels in bottom line. The wraparound channel connects level 1 nodes to level 0 nodes. Note that the wraparound channel is the only channel that does not have two virtual channel to it. The dependency graph for the DRVI is $D(VC, E)$, where $E = \{(vc_{i,L}, vc_{\overline{i+1},L})\}$, except for $i = n-1$ in which case the corresponding dependency edge is $(vc_{n-1,1}, vc_{0,0})$. In the following it is proved that DRVI is acyclic.

THEOREM 9.3 *The channel dependency graph in virtual interconnection DRVI is acyclic.*
Proof: If the destination of a packet originated at i is j and $j > i$, then $L = 0$. Otherwise $L = 1$, the packet travels in level 1, crosses over the wraparound channel, travels in level 0, and reaches to j (refer Figure 9.5). Let a flit arriving from channel $vc_{i,L}$ will acquire $vc_{\overline{i+1},L}$ next. We now attempt to number the channels as per the dependency. For simplicity let

$$\theta(vc_{i,L}) = (1-L)n + i \tag{9.1}$$

where $L = 1, 0$. This way all channels in the virtual network are numbered in the ascending order from 0 to $2n-2$. Hence from the Definition 9.6, the channel dependency graph D for DRVI is acyclic. ♠

We now consider a routing that uses the above dependencies so that the routing is deadlock-free. Let the shortest path length from node pn_i to node pn_j be represented by $y[+1]$. The following is the routing algorithm that avoids deadlocks.

ALGORITHM 9.1 *Routing in DR under virtual channels without deadlocks.*

- *If $pn_i + Y = 0$ for some $Y \leq y$, use virtual channels in level 1 from pn_i to pn_{n-1}, later travel on the wraparound channel from node pn_{n-1} and finally use virtual channels in level 0 from pn_0 to pn_j.*

  It means the packets which need wraparound channels start routing in virtual channels *in level* 1 *and later those in level* 0.

- *If* $pn_i + Y \neq 0$ *for any* $Y \leq y$, *use virtual channels in level* 0 *to transfer packets. It means the packets that do not need the wraparound channel stay in level* 0 *and route to destination.* ♠

9.8 Undirected ring

This network is an undirected ring $UR(PN, PC)$. Let the physical nodes be represented by pn_0, pn_1, $\ldots pn_{n-1}$. We distinguish the links pn_i to $pn_{\overline{i+1}}$ and to $pn_{\overline{i-1}}$ by the usual notation defined in Section 2.2 by $[+1]$ and $[-1]$ links respectively, where $\overline{j} = j \ (mod\ n)$. The channels of $[+1]$ links are numbered $pc_{i,+1}$ where i is the physical node position from where flits enter and the $+1$ indicates the direction of the channel. Similarly the $[-1]$ links are numbered (refer Figure 9.6). We can consider the UR as two independent networks, the UR_{+1} and and UR_{-1}

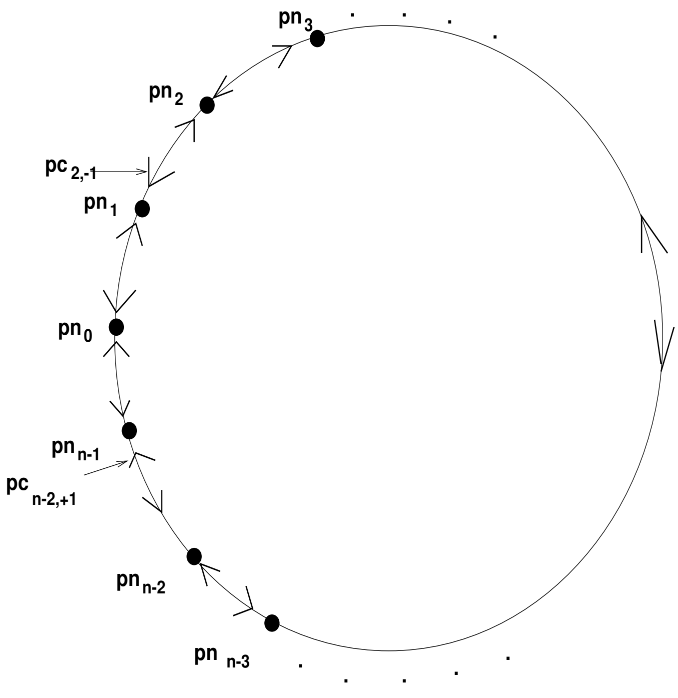


Figure 9.6: Undirected ring physical interconnection

each having the $[+1]$ and $[-1]$ links respectively. The number of $[+1]$ links ($[-1]$ links) between a source-destination pair is represented by y (or z). As per the Theorem 9.2 both the UR_{+1} and UR_{-1} have cyclic channel dependencies. However In the following theorem we identify the channels that do not form cyclic dependency among the channels of UR.

THEOREM 9.4 *In the minimal routing in* UR, *there exist no dependency between* $[+1]$ *and*

[−1] channels.

Proof: The shortest path between a source destination pair in UR consists of either only [+1] or only [−1] links. Thus in minimal routing, a packet routed from a source will be traveling either only in [+1] links or only in [−1] links. So the packets that travel in [+1] links do not depend for the [−1] channels implying that there exist no dependency between [+1] and [−1] channels in UR. ♠

Hence the channels between UR_{+1} and UR_{-1} do not have any dependencies. It can also be noted that the cyclic dependency of channels in [+1] (similarly in [−1]) links is due to the existence of wraparound channel.

The wraparound channel in [+1] link direction is as defined in Definition 9.7. We now define the wraparound channels in [−1] link direction. If z is the shortest path length from node pn_i to node pn_j and $(i + Z)\ mod\ n = n - 1$ for some $Z \leq z$, then we say that the path traverses through the wraparound channel. Thus the following definition follows.

DEFINITION 9.8 *The channels from pn_{n-1} to pn_0 and pn_0 to pn_{n-1} are the wraparound channels in UR.* ♠

The channel dependencies in UR_{+1} (similarly UR_{-1}) can be eliminated with virtual channels as in DR. The respective virtual interconnections for UR_{+1} and UR_{-1} are given by $URVI_{+1}$ and $URVI_{-1}$) and shown in Figure 9.7.

We number each virtual node with a two letter suffix: the first suffix indicating node position and the second indicating the level in which the node is mapped to. For example $pn_{12,0}$ is virtual node numbered 12 in level 0. Similarly the channels are identified with a three letter suffix, the first suffix indicates the direction of the channel (either +1 or −1), the second suffix indicates the node position from where flits enter the channel and the third indicates the level number (1 or 0). For example $vc_{+1,12,0}$ is a virtual channel in level 0 originating from node position 12 in the direction of [+1] links.

We now define the $nmap_{+1}$, $cmap_{+1}$ for $URVI_{+1}$ and similarly for $URVI_{-1}$ in the following.

$$\begin{aligned}
nmap_{+1} &: \ pn_{i,L} \rightarrow \ pn_i \\
cmap_{+1} &: \ vc_{+1,i,L} \rightarrow \ vc_i \\
&: \ Undefined\ for\ i = n-1\ and\ L = 0 \\
nmap_{-1} &: \ pn_{i,L} \rightarrow \ pn_i \\
cmap_{-1} &: \ vc_{-1,i,L} \rightarrow \ vc_i \\
&: \ Undefined\ for\ i = 0\ and\ L = 0
\end{aligned}$$

The above virtual interconnection definitions are used in the following to describe a deadlock-free routing. The channel dependency graphs for both the links are $D_{+1}(VC_{+1}, E_{+1})$ and $D_{-1}(VC_{-1}, E_{-1})$, where E_{+1} and E_{-1} are defined from the respective dependencies in [+1]

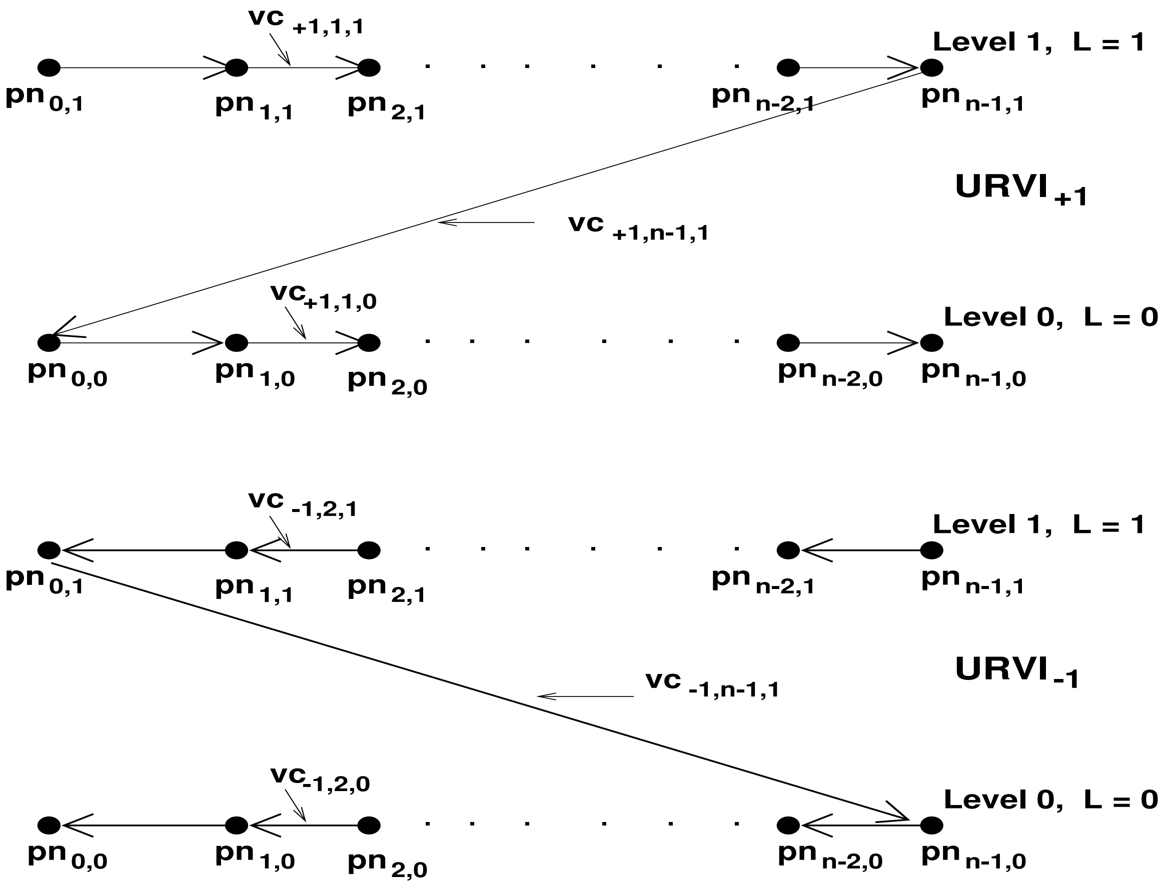


Figure 9.7: Virtual interconnection in an undirected ring

and $[-1]$ links. The edge set of dependency graph D_{+1} is $E_{+1} = (vc_{+1,i,L},\ vc_{+1,\overline{i+1},L})$ except for $i = n-1$, $L = 1$ in which case the corresponding dependency edge is $(vc_{+1,n-1,1}, vc_{+1,0,0})$. Similarly the edge set of dependency graph D_{-1} is $E_{-1} = (vc_{-1,i,L},\ vc_{-1,\overline{i-1},L})$ except for $i = 0$, $L = 1$ in which case the corresponding dependency edge is $(vc_{-1,0,1}, vc_{-1,n-1,0})$. In the following we prove that $URVI_{+1}$ and $URVI_{-1}$ have no cyclic channel dependencies, given the channel dependencies by E_{+1} and E_{-1}.

THEOREM 9.5 *The channel dependency graph in the virtual interconnection $URVI_{+1}$ (and similarly $URVI_{-1}$) is acyclic.*

Proof: The proof that channel dependency graph in $URVI_{+1}$ is acyclic is similar to the numbering scheme of (9.1) in Theorem 9.3. We now prove that D_{-1} of $URVI_{-1}$ is also acyclic using a similar numbering scheme for the channels in $URVI_{-1}$. For simplicity the channels in $URVI_{-1}$ are numbered

$$\theta(vc_{-1,i,L}) \quad = \quad (2-L)n - i - 1 \tag{9.2}$$

in the order of channel direction. Thus we get the channel numbering from 0 to $2n-2$, starting *from channel $vc_{-1,n-1,1}$ to channel $vc_{-1,1,0}$. Thus the dependency graph D_{-1} for $URVI_{-1}$ is acyclic.* ♠

Hence $URVI$ has no cyclic channel dependencies. We give below a deadlock-free routing in UR using virtual channels of $URVI$. Let pn_i be the source and pn_j be the destination of a packet. Also assume that the shortest path is either y number of $[+1]$ links or z number of $[-1]$ links.

ALGORITHM 9.2 *Routing in UR under virtual channels without deadlock.*

- *If the shortest path is $y[+1]$*
  If $pn_i + Y \neq 0$ for any $Y \leq y$, use virtual channels of $URVI_{+1}$ in level 0 to route to destination
  If $pn_i + Y = 0$ for some $Y \leq y$, use virtual channels of $URVI_{+1}$ in level 1 from source to node $pn_{n-1,1}$, later on the wraparound channel $vc_{+1,n-1,1}$ to node $pn_{0,0}$ and the remaining path in level 0 to the destination.

- *If the shortest path is $z[-1]$*
  If $pn_i + Z \neq n - 1$ for any $Z \leq z$, use virtual channels of $URVI_{-1}$ in level 0 to route to destination
  If $pn_i + Z = n - 1$ for some $Z \leq z$, use virtual channels of $URVI_{-1}$ in level 1 to route from source to node $pn_{0,1}$, later on the wraparound channel $vc_{-1,0,1}$ and the remaining path in level 0 to destination ♠

9.9 Undirected MCDL

Here we provide a deadlock-free minimal routing in $MCDL(PN, PC)$ using virtual channels. The PN is a set of physical nodes labelled $\{pn_0, pn_1, \dots pn_{n-1}\}$. The $PC = \{(pn_i, pn_{\overline{i\pm1}})), (pn_i, pn_{\overline{i\pm h}})\}$, where $\overline{j} = j$ $(mod$ n),. and h is the hop size. Consider that $g = GCD(n, h)$. The physical channel connections $(pn_i, pn_{\overline{i\pm1}})$ form a loop ring and $(pn_i, pn_{\overline{i\pm h}})$ form g hop rings. We distinguish the four types of channels at a node pn_i with a two-subscripted identifier to pc: the first subscript identifying the direction and the second identifying the node from where flits enter the channel. For example $pc_{-h,j}$ is a channel from node pn_j to node $pn_{\overline{j-h}}$.

Thus in MCDL there are g undirected hop rings and one undirected loop ring or $(2g + 2)$ directed rings. Let the physical interconnection of the $2g$ directed hop rings be called $MCDL_{+h}$ and $MCDL_{-h}$ networks having the $[+h]$ and $[-h]$ links respectively. Similarly the 2 directed rings are $MCDL_{+1}$ and $MCDL_{-1}$ with $[+1]$ and $[-1]$ links respectively. The $MCDL_{+h}$ $MCDL_{-h}$ $MCDL_{+1}$ and $MCDL_{-1}$ networks do have cyclic channel dependencies which can be proved similar to Theorem 9.2.

As in UR and DR, the MCDL also has wraparound channels for each hop ring and loop ring. They are in the following definition.

DEFINITION 9.9 *The following are the wraparound channels in MCDL.*

1. *The directed channel from node $n-1$ and 0 i.e., $pc_{+1,n-1}$ is the wraparound channel for the $[+1]$ loop.*

2. *The directed channel from node 0 and $n-1$ i.e., $pc_{-1,0}$ is the wraparound channel for the $[-1]$ loop.*

3. *There exist g number of hop rings, labelled $0, 1, \ldots (g-1)$. Each hop ring has a wraparound channel in both the $[+h]$ and $[-h]$ directions.*
   The i^{th} $[+h]$ hop ring wraparound channel is from node $(n-h+i_g)$ to node i_g i.e., $pc_{+h,n-h+i_g}$ where $i_g = 0, 1, \ldots (g-1)$.
   The i^{th} $[-h]$ hop ring wraparound channel is from node i_g to node $(n-h+i_g)$ i,e., pc_{-h,i_g} where $i_g = 0, 1, \ldots (g-1)$. ♠

To eliminate the channel dependencies in MCDL due to the wraparound channels, we introduce virtual channels like in UR and DR. Each physical channel has two multiplexed virtual channels. The complete set virtual channels at a node i is shown in Figure 9.8 where each unidirectional physical channel multiplexes two virtual channels, except the wraparound chan-

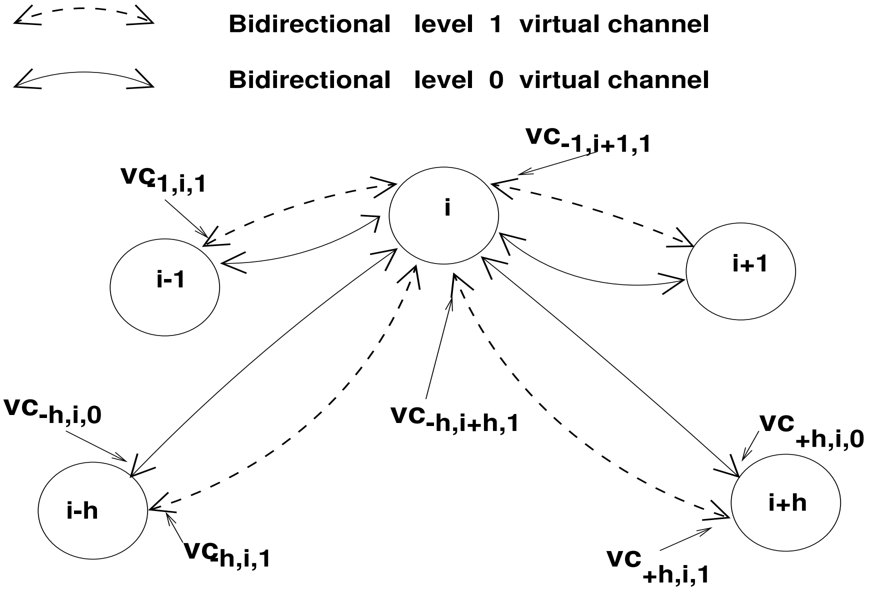


Figure 9.8: Virtual channels at node i

nels. The channels are in one of the two levels labelled $0, 1$. Each virtual channel is indexed with three subscripts: the first indicates hop or loop with appropriate direction, the second indicates the node number from where flits enter and the third indicates the level number of the virtual channel (0 to 1). For example $vc_{+h,23,1}$ is a virtual channel at level 1 on the $[+h]$ link from node number 23. The *cmap* and *nmap* are defined below for $L = 0, 1$.

$$\begin{aligned}
nmap &: pn_{i,L} \rightarrow pn_i \\
cmap_{+1} &: vc_{+1,i,L} \rightarrow pc_i \\
cmap_{-1} &: vc_{-1,i,L} \rightarrow pc_i \\
cmap_{+h} &: vc_{+h,i,L} \rightarrow pc_i \\
cmap_{-h} &: vc_{-h,i,L} \rightarrow pc_i
\end{aligned}$$

The virtual interconnections are $MCDLVI_{+h}$, $MCDLVI_{-h}$, $MCDLVI_{+1}$ and $MCDLVI_{-1}$ do not have any cyclic channel dependencies This can be proved as in Theorem 9.3.

In Section 5.2, we introduced two types switching for routing messages: HTL and LTH switching models. Recall from equation (2.1) that minimal routing is in only one of the four link combinations. So we group the $(2g + 2)$ virtual interconnection networks based on the link combination to which the shortest path belongs. Thus we obtain four virtual interconnections, namely: $MCDLVI_{+h,+1}$, $MCDLVI_{+h,-1}$, $MCDLVI_{-h,+1}$ and $MCDLVI_{-h,-1}$, shown in Figure 9.9 to 9.12. We use the terminology of Section 2.2, to denote the number of $[+h]$, $[-h]$, $[+1]$ and $[-1]$ channels in minimal routing between a pair of nodes by $w,\ x,\ y,\ z$. In the following, we give sequence of physical channels used from source node s to a destination in the minimal routing in both the HTL and LTH routings.

$$\begin{array}{llll}
R_{HTL}: & \text{In } (+h,+1): & pc_{+h,s+ih} & i = 0 \ldots w-1 \\
 & & pc_{+1,s+wh+j} & j = 0 \ldots y-1 \\
 & \text{In } (+h,-1): & pc_{+h,s+ih} & i = 0 \ldots w-1 \\
 & & pc_{-1,s+wh-j} & j = 0 \ldots z-1 \\
 & \text{In } (-h,+1): & pc_{-h,s-ih} & i = 0 \ldots x-1 \\
 & & pc_{+1,s+wh+j} & j = 0 \ldots y-1 \\
 & \text{In } (-h,-1): & pc_{-h,s-ih} & i = 0 \ldots x-1 \\
 & & pc_{-1,s+wh-j} & j = 0 \ldots z-1 \\
R_{LTH}: & \text{In } (+h,+1): & pc_{+1,s+i} & i = 0 \ldots y-1 \\
 & & pc_{+h,s+y+jh} & j = 0 \ldots w-1 \\
 & \text{In } (+h,-1): & pc_{-1,s-i} & i = 0 \ldots x-1 \\
 & & pc_{+h,s-x-jh} & j = 0 \ldots w-1 \\
 & \text{In } (-h,+1): & pc_{+1,s+i} & i = 0 \ldots y-1 \\
 & & pc_{-h,s+y-jh} & j = 0 \ldots x-1 \\
 & \text{In } (-h,-1): & pc_{-1,s-i} & i = 0 \ldots z-1 \\
 & & pc_{-h,s-z-jh} & j = 0 \ldots x-1
\end{array}$$

In the following two theorems we identify the channels that do not form cyclic dependencies among them.

THEOREM 9.6 *In minimal routing under R_{HTL} (similarly under R_{LTH}), there exist no cyclic dependency between hop and loop channels.*
Proof: Consider R_{HTL}. The shortest path lies in one of the four link combinations. As per the routing of the packets, they first travel on the hop ring and later travel on the loop ring. Thus there exist a dependency from hop ring channels to loop ring channels but not in the reverse. Hence cyclic dependency is not possible between the hop and loop channels. Similarly in R_{LTH} also cyclic dependency is not possible between the hop and loop channels. ♠

THEOREM 9.7 *In minimal routing under R_{HTL} (similarly under R_{LTH}) there exist no cyclic dependency between*

1. *$[+h]$ and $[-h]$ channels*
2. *$[+1]$ and $[-1]$ channels*

Proof: Similar to the proof given in Theorem 9.4 for DR.

THEOREM 9.8 *In HTL (or similarly in LTH), The virtual interconnections $MCDLVI_{+h,+1}$, $MCDLVI_{+h,-1}$, $MCDLVI_{-h,+1}$ and $MCDLVI_{-h,-1}$ are deadlock-free.*
Proof: Consider the virtual interconnection $MCDLVI_{+h,+1}$ in HTL routing (or similarly in LTH routing) for the $(+h,+1)$ link combination of the shortest path. This $MCDLVI_{+h,+1}$ can be decomposed into g number of hop rings with $[+h]$ links and one loop with $[+1]$ links. Each of these $(g+1)$ rings has a virtual interconnection similar to the $DRVI_{+1}$ in Figure 9.5. Using the numbering scheme of channels as in (9.1) for the $(g+1)$ virtual interconnections, it can be proved that the channel dependencies in them are cyclic.

The new additional channel dependencies for the $MCDLVI_{+h,+1}$ are between $[+h]$ and $[+1]$ channels. It is proved in Theorem 9.6 that this additional dependency between $[+h]$ and $[+1]$ channels is not cyclic. Hence the channel dependencies in $MCDLVI_{+h,+1}$ are acyclic. Similarly the channel dependencies in the other three virtual interconnections $MCDLVI_{+h,-1}$, $MCDLVI_{-h,+1}$ and $MCDLVI_{-h,-1}$ can also to be proved to be acyclic. Hence the MCDL with virtual interconnections is deadlock-free for minimal HTL routing (similarly for minimal LTH routing). ♠

The restrictions on the virtual channels for routing the packets are same as those in DR and UR regarding the relative positions of source and destination addresses. In minimal routing of HTL or LTH, since packets travel through hop and loop rings, we have to consider the possibility of crossing the wraparound channels of both the hop and loop rings. We explain it by considering the HTL. Let w and y are the $[+h]$ and $[+1]$ links in a shortest path from i to j. Refer the channel sequences in HTL given in the above. After w number of hop traversals the node reached is $i + wh \ (mod\ n)$. If $(i + wh \ (mod\ n) < i$, then the path contains the wraparound channel of the hop. Otherwise the path does not contain the wraparound channel. Similarly when the packet moves from node $i + wh \ (mod\ n)$ to j for loop traversals, then if $j < i + wh + y \ (mod\ n)$ then the path contains the wraparound channel of the loop. Otherwise the loop path does not contain the loop wraparound channel. Accordingly, the levels of the virtual interconnections are chosen similar to in the protocols of DRVI and URVI routings given Algorithms 9.1 and 9.2. Since these routings use the acyclic channels of the corresponding virtual interconnections, they are deadlock-free. The routing is given in Algorithm 9.3.

ALGORITHM 9.3 *A minimal non-adaptive wormhole routing algorithm in MCDL.*

- *Compute shortest path and obtain w, x, y, z accordingly, (refer Section 3.3 and 3.4) from source node s to destination node d.*
  In the routing table below $0 < W, X, Y, Z \leq w, x, y, z$ respectively and $0 \leq i_g < g$.
  $g = 0, 1, 2 \dots (g-1)$.
- *The steps in routing in the other link combinations are similar to those elaborated for $(+h, +1)$.*

Routing Table in (+h,+1) under HTL

Shortest path link	*Routing type*	*wraparound condition*	*Routing steps (channel sequence)*
$(+h, +1)$	*HTL*	$s + W \cdot h = i_g$	$vc_{+h,s,1}$ *to* $vc_{+h,s+(W-1)h,1}$
			$vc_{+h,s+W \cdot h,0}$ *to* $vc_{+h,s+(w-1)h,0}$
		$s + W \cdot h \neq i_g$	$vc_{+h,s,0}$ *to* $vc_{+h,s+(w-1)h,0}$
		$s + w \cdot h + Y = 0$	$vc_{+1,s+wh,1}$ *to* $vc_{+1,s+wh+Y-1,1}$
			$vc_{+1,s+wh+Y,0}$ *to* $vc_{+1,s+wh+y-1,0}$
		$s + w \cdot h + Y \neq 0$	$vc_{+1,s+wh,0}$ *to* $vc_{+1,s+wh+y-1,0}$

♠

9.10 Summary

In this chapter we have given deadlock-free wormhole routing algorithms in MCDL with virtual channels. We have shown that the HTL and LTH routings are deadlock-free with only two virtual channels for every physical channel in MCDL. Each of the virtual channels is in a *level*. We defined a wraparound channel and this channel is used to link the two levels.
Consider a routing where packets travel in a *convoluted* fashion (*i.e*, neither HTL nor LTH). In other words packets select en route an output channel at a node from one of the $[+h]$, $[-h]$, $[+1]$ or $[-1]$ channels randomly. Such a routing also called *adaptive routing* and can be either minimal or non-minimal.

In a *minimal adaptive routing*, there exist cyclic channel dependency between hop and loop channels. So Theorem 9.7 is not true in this case and there exist multilevel channel dependency. Hence the virtual interconnections proposed in Figure 9.9 to 9.12 are not deadlock-free. Moreover if the adaptive routing is non-minimal, then additionally we have cyclic dependency among the hop and loop channels. In this case also the virtual interconnections are not deadlock-free. However, for a non-minimal non-adaptive routing, the virtual interconnection in is deadlock-free provided the packets do not travel the wraparound channels more than once in their non-minimal path.

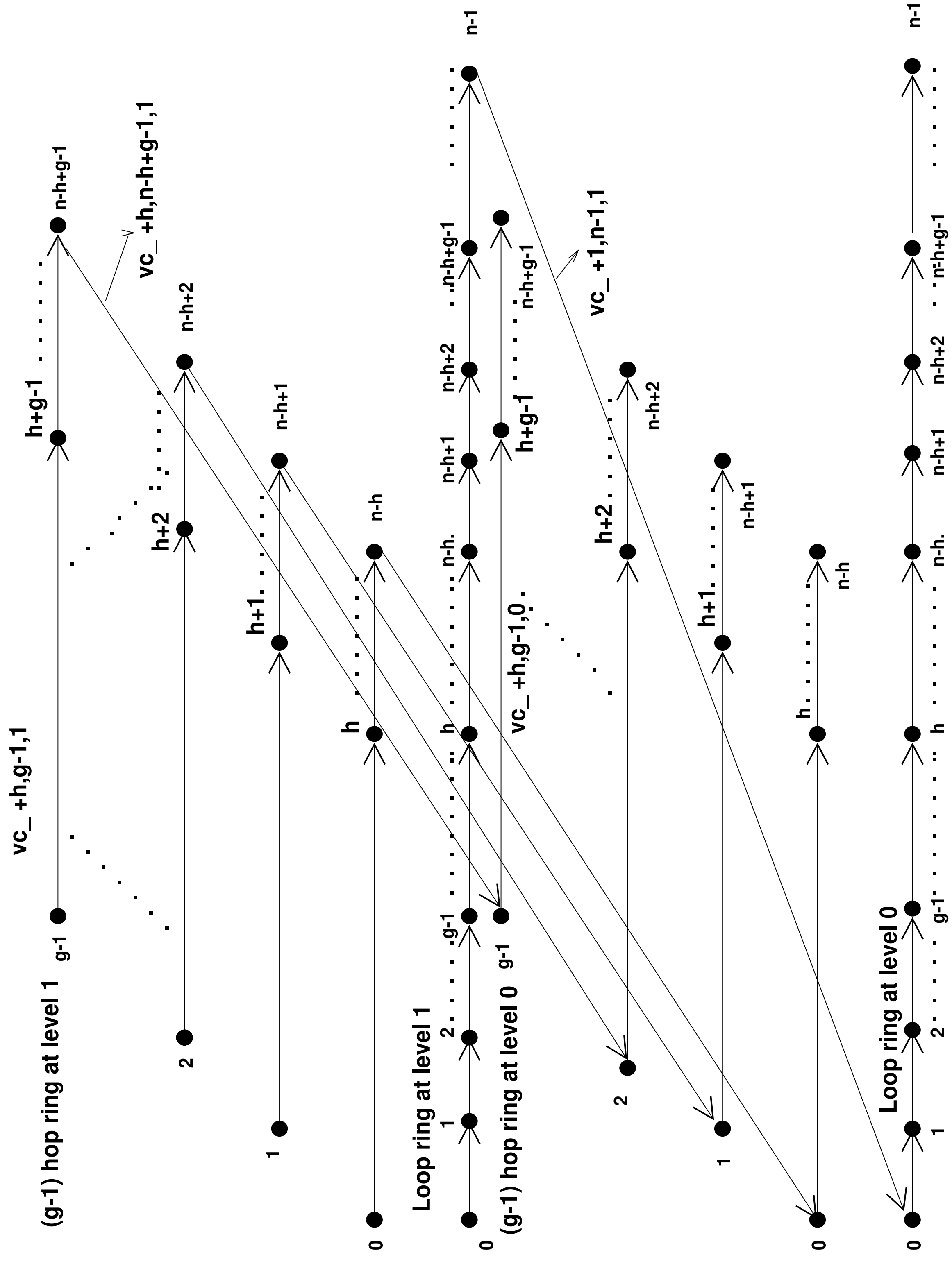


Figure 9.9: $MCDLVI_{+h,+1}$ virtual channels for $(+h,+1)$ physical interconnection

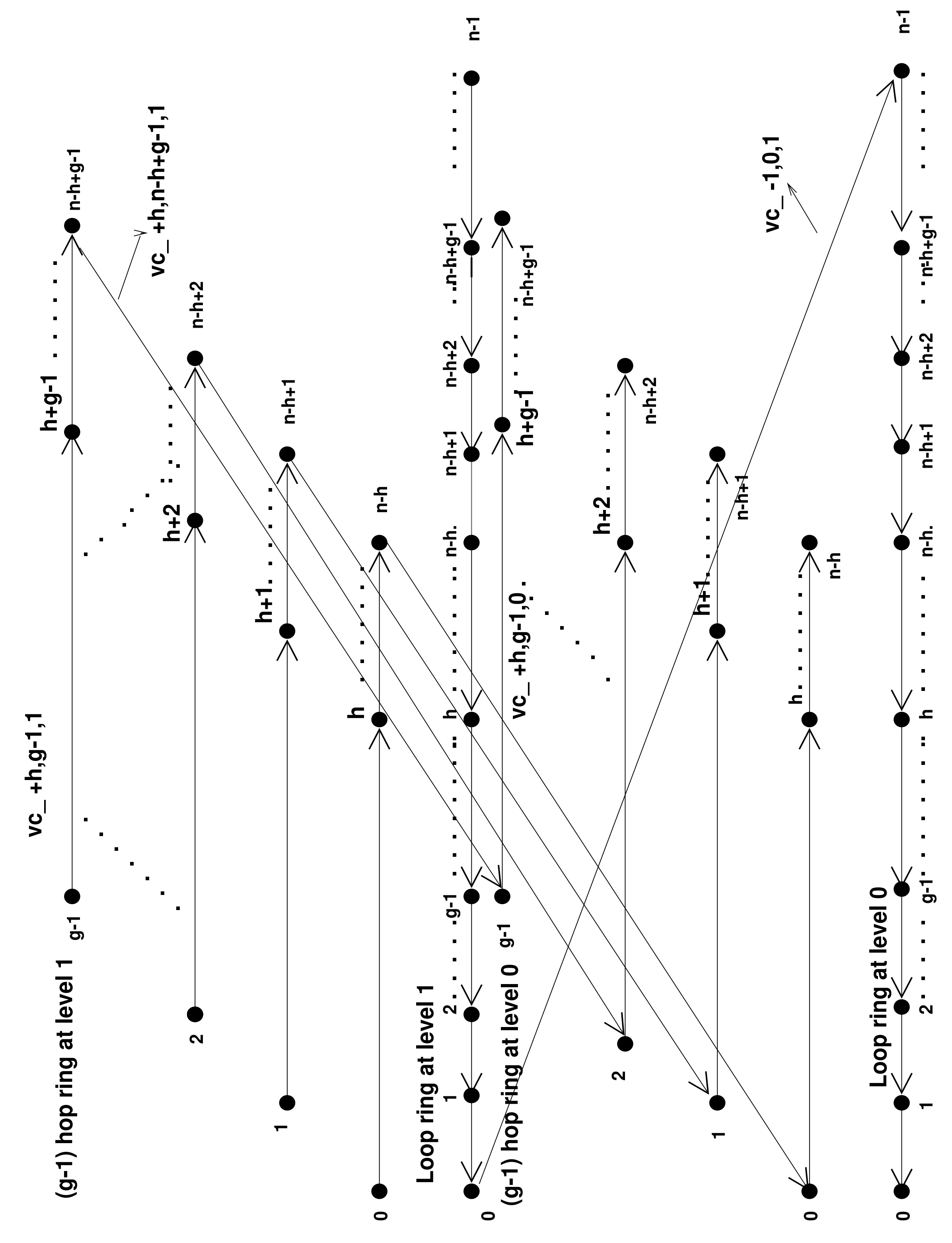


Figure 9.10: $MCDLVI_{+h,-1}$ virtual channels for $(+h, -1)$ physical interconnection

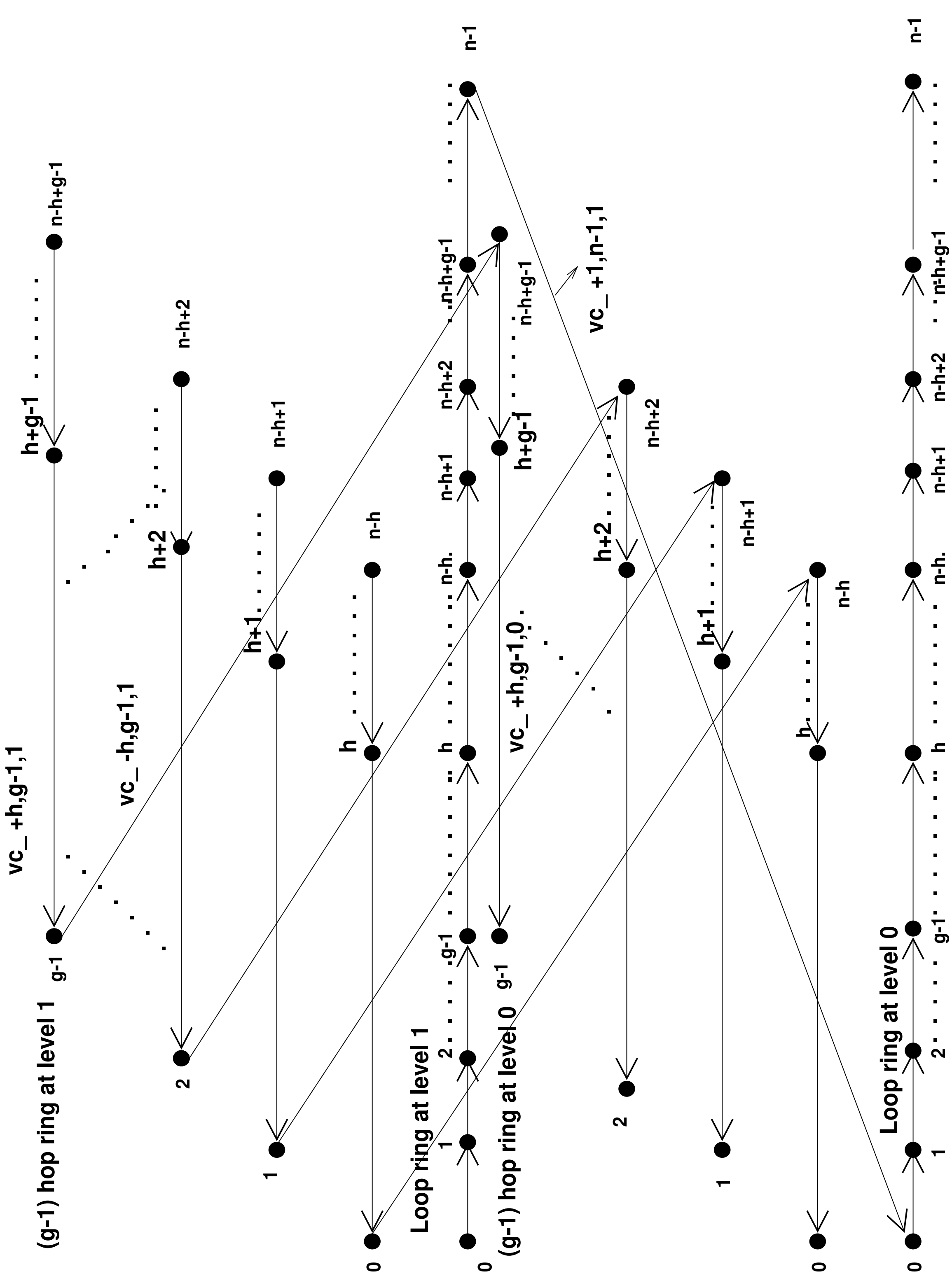


Figure 9.11: $MCDLVI_{-h,+1}$ virtual channels for $(-h,+1)$ physical interconnection

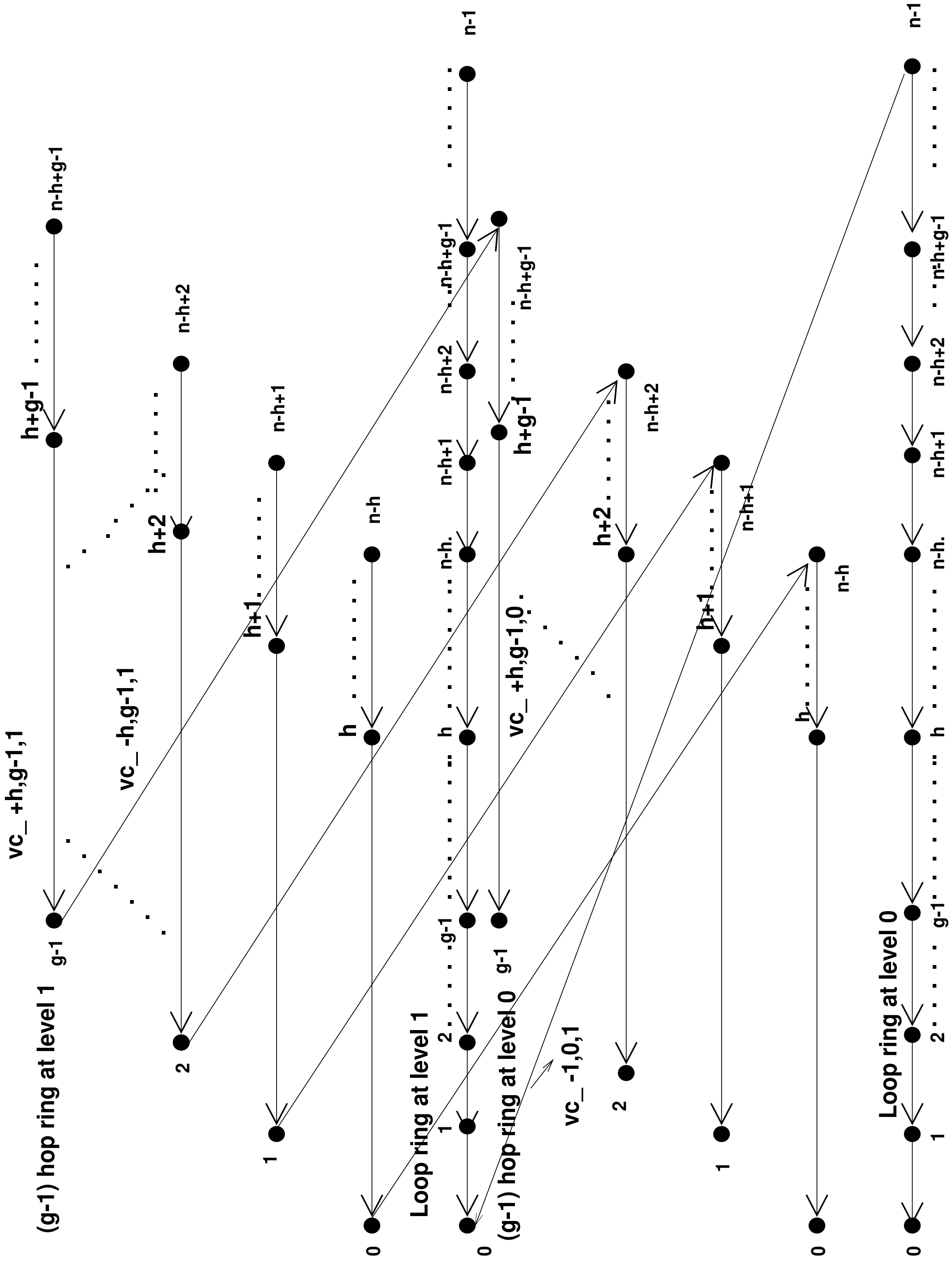


Figure 9.12: $MCDLVI_{-h,-1}$ virtual channels for $(-h,-1)$ physical interconnection

Chapter 10

CONCLUSIONS

In this thesis we have presented several results concerning the MCDL networks as an interconnection topology for *shared medium* and *direct network* applications. We believe that these results extend and also contribute to our knowledge of the MCDL network as an interconnection topology. We applied concepts of *modular arithmetic* and *linear congruences* to develop a faster algorithm for shortest path computation in MCDL. The ideas of queueing theory techniques and multiple token media access protocols are used in finding packet transfer delays in MCDL and to make a comparative study with single ring and multiple ring token protocols. The fault tolerant routing and deadlock-free wormhole techniques given in this thesis consider a new direction towards addressing the respective problems.

The key set of issues addressed include *shortest path, fault tolerant routing, deadlock-free wormhole routing* and *multitoken media access protocol.*

10.1 Summary of results

The MCDL of n nodes is a ring like topology in which node i is connected to nodes $(i \pm 1)$ $mod\ n$ forming a *loop* ring and to nodes $(i \pm h)$ $mod\ n$ forming g number of *hop* rings, where $g = GCD(n, h)$ and h is called *hop size.*

A closed form expression for finding the shortest path in MCDL is an unsolved problem. Algorithms exist that find the shortest path in time $O(\delta)$ where δ is diameter of the MCDL [MS95]. The shortest path algorithm given by us has $O(\frac{h}{g}+log\ h)$ run time which is independent of diameter. We used number theoretic properties in this algorithm.

Based on this shortest path algorithm, we have studied graph properties of the MCDL network such as *diameter, average inter node distance* and their optimal values and observed a non-monotonic variation of these parameters with h and n. We have also given a comparison of MCDL network with other topologies such as *k-ary n-cube, mesh* and *torus* networks. For example, the diameter of a mesh having the same number of nodes as MCDL is more than double for any number of nodes. Whereas the torus has a diameter that is more by 1.5 times than MCDL of equal number of nodes. However the k-ary n-cube has less diameter than MCDL. But the degree of a k-ary n-cube is $2 \cdot n$ while the degree of an MCDL is only 4. Thus the construction costs of MCDL are less than a k-ary n-cube for the same number of nodes .

Based on the *token ring media access* control [SS85], we have developed a multitoken media access control in MCDL. Multitoken media access control was also analysed with *multiple rings* in [BGY89]. Their results indicate that the delays in multiple rings are less than single ring [SS85]. But this gain in delay is not seen for all values of number of rings [VR94]. As the number of rings increases beyond a certain number, as given in [VR94], the delays also start increasing. This is due to token *coalescence* and occurs because of the media access protocol. In multiple rings, the media access protocol uses only one of the several tokens. In other words if a node possesses a token for transmitting a packet and in case any of the other tokens also reach this node at that time, then those tokens are unused and are simply passed on to the nodes down the links. This strategy avoids any buffer conflicts in accessing the single buffer at a node. Thus even though the token inter arrival time reduces due to multiple tokens, because of the above strategy of media access, tokens are likely to coalesce at nodes resulting in a performance almost equal to that of single ring. In MCDL at any node at most two tokens may arrive the same time. MCDL avoids token coalescence for the simple reason that packets arriving a node are queued in different buffers for the two rings. The routing algorithm decides which buffer the packet enters.

Moreover, the shortest path result has given rise to three broad routing schemes: *hop to loop (HTL), loop to hop (LTH)* and *no-switching (NOS).* In these routings the traffic can be classified as hop, loop and switch over types based on the destination. Separate buffers have been allocated to these types of traffic. Except the switch over, the hop and loop traffics use the corresponding rings that are served by the tokens of their own. However the switch over traffic uses both the hop and loop rings. This introduces an extra stage of queueing. We found in our analysis that this extra stage queueing is more dominant when hop sizes are large. Accordingly, even though shortest paths are used, the delays of MCDL are no better than multiple rings even at moderately small hop sizes.

In the analysis it was shown that the switch over traffic in HTL and LTH routing cases is approximately Poisson. We analysed the packet delays in the three routing schemes. We also analysed the effect of packet length and hop size on the delay. All our analytical results are supported with a discrete event simulator designed for this purpose.

We have also studied MCDL as a direct network interconnection topology for multicomputer applications. The most commonly used communication protocol in multicomputer direct networks is *wormhole* routing. The advantage of wormhole routing is that it has a communication latency that is independent of inter node distances. Also packets are split into small sized flits before transmission thus requiring little buffer management and small storage in the nodes. But a common problem in wormhole routing is network blocking and deadlock. The deadlocks may be avoided in wormhole routing by *virtual channels.* Study of deadlock-free wormhole routing in k-ary n-cube mesh and tori are found in literature [DS87, LH91, DA93, BC96]. But we found little work in MCDL for a deadlock-free wormhole routing and its applicability in direct network interconnection topologies.

We designed algorithms for a deadlock-free wormhole routing in MCDL using the virtual channels along the lines of [LH91]. Our work presents a subtle difference from the treatment given in earlier works in [BC96, DA93, LH91]. In [BC96, DA93], a routing function was defined for a virtual interconnection network having virtual channels and then a channel dependency graph was derived from this routing by using it as an implicit definition of all possible connections [DS87]. Then it was proved that if the channel dependency graph is acyclic, the routing function is deadlock-free. In contrast, in [LH91] the physical interconnection network (without virtual channels) is used to define what connections are possible and from that the channel interdependencies are derived. Then for any cyclic channel dependencies *virtual channels* were introduced to remove those cyclic dependencies *acyclic.* Now any routing function that uses only these virtual channels is deadlock-free.

But in our work that is also in the lines of [LH91], we keep the construction of channel dependency graphs simple by hierarchical decomposition of the physical interconnection. In addition we show that if a routing function adds new dependencies (not already in the set of dependencies), it may still be possible to easily argue the deadlock freedom. Thus, instead of showing each new routing function deadlock-free, the virtual interconnection network is shown to be deadlock-free for a given channel dependency graph and any routing function that obeys these restrictions of this channel dependency graph is also deadlock-free.

With this background, we consider *directed ring*, and *undirected ring* and extend results for these networks to *MCDL* topologies. These networks are first shown to be having cyclic channel dependencies. So virtual channels are introduced to remove those cyclic dependencies. Then the *hop to loop* and , *loop to hop* routings that use these virtual channels are shown to be deadlock-free, even though they introduce new channel dependencies.

Lastly we consider the problem of routing packets under faults in MCDL. In our work we primarily consider two cases of network state under faults. They are *global knowledge* of faults and *neighbourhood knowledge* of faults. In the former case, the address of the faulty node or edge is available at all other nodes in the network. In the latter case they are known only to its adjacent nodes. We show how to efficiently route a packet in a faulty MCDL under these two assumptions of faults. We also present conditions under which the shortest path length increases in both the cases of fault knowledge, and show that it is no more than two.

10.2 Future work

The general approaches undertaken in this work may be used to further investigate some other issues in MCDL interconnection network. Specific issues include:

1. Investigation of other ring protocols such as: *slotted* and *register insertion* and comparison with single ring multiple ring access protocols.
2. Investigation of fault diameter under multiple node/edge faults.
3. Investigation and performance of fault tolerant deadlock-free adaptive and non-adaptive wormhole routing schemes in direct network applications.

Techniques used in our work to obtain deadlock-free routing for MCDL may be adaptable to construct deadlock-free interconnection network with virtual channels for other similar networks such as: k-ary n-cube, mesh, torus *etc*,.

A generalisation of the MCDL is called *circulant* interconnection network $G(n;\ s_1,\ s_2, \ldots s_k)$. In a circulant, a node i is connected to nodes $(i \pm s_p)\ mod\ n$, $1 \leq p \leq k$. Our work may be extendable to the investigation and study this class of networks to compute shortest path, optimal diameters and other parameters.

In chapter 5 through 7 it is observed that at higher hop sizes the loop traffic increases and consequently the weighted average delay in multiple token MCDL also increases. Thus the delay is limited by loop token service time. In view of this, we propose to invetigate and study mixed protocol for media access in MCDL specifically *register insertion* with *destination removal* to increase service rate of loop traffic.

Appendix A

COMMUNICATIONS RELATED TO THE WORK

1. N. Chalamaiah and R. Badrinath, Communication Primitives in Ring Connected Multicomputers, Proceeding of IEEE National Conference on INTERNET for INDIA, Hyderabad, India, Dec,11-13,1997, IEEE Hyderabad chapter, pages: 17-20.

2. N. Chalamaiah and R. Badrinath, Analysis of Multiconnected Loop Topologies for Interconnection Networks, *Proceedings of National Systems Conference-97 on Integrated Systems with Multiple Technologies, DRDO, RCI, Hyderabad, India,* Jan,22-24,1998, Allied publishers, pages: 327-332.

3. N. Chalamaiah and R. Badrinath, A Multitoken Distributed Loop for Local Area Networks, *Proceedings of National Systems Conference-97 on Integrated Systems with Multiple Technologies, DRDO, RCI, Hyderabad, India,* Jan,22-24,1998, Allied publishers, pages: 333-338.

4. N. Chalamaiah and R. Badrinath, Finding Shortest Paths in Distributed Loop Networks, *Information Processing Letters*: 67(3), 1998, 157-161.

5. N. Chalamaiah and R. Badrinath, Analysis of Multiconnected Loop Topologies for Interconnection Networks, *Proceedings of 2nd European Parallel and distributed Systems Conference Euro-PDS98, Vienna,Austria,* Jul, 1-3,1998,

6. N. Chalamaiah and R. Badrinath, Multiple Token Distributed Loop Local Area Networks: Analysis, Proceedings of *5th International Conference on High Performance Computing, Chennai, India,* Dec,17-20,1998, IEEE Computer Society press, pages: 400-407.

7. N. Chalamaiah and R. Badrinath, Multiconnected Distributed Loop Interconnection Networks, *Proceedings of International Conference on Information Technology*, CIT-98, Bhubaneswar,India, Dec, 21-23,1998, TMH Publishers, pages: 181-184.

8. N. Chalamaiah and R. Badrinath, Modelling and Analysis of Multitoken Multiconnected Loop Networks, Proceedings of *International Conference on Computers and Information Technology, Dhaka,Bangladesh,* Dec,18-20,1998, pages: 2-7.

9. N. Chalamaiah and R. Badrinath, Modelling and Analysis of Multitoken Multiple Loop Networks, *Proceedings of International Conference on Networking India and The World, Ahmedabad,India.* Dec, 9–12,1998, IEEE Gujarat section, pages: 49-55.

10. N. Chalamaiah and R. Badrinath, Analysis Modelling and Simulation of Multitoken Multiconnected Distributed Loop Local Area Networks, *Proceedings of the 5th National Conference on Communications NCC-99*, Indian Institute of Technology, Kharagpur, India, Allied Publishers, Jan,30-31,1999, pages: 417-425.

11. N. Chalamaiah and R. Badrinath, Analysis Modelling and Simulation of Multitoken Multiconnected Distributed Loop Local Area Networks, *Computer Communications*, communicated.

Appendix B

MODULAR ARITHMETIC

Informally we can think of *modular arithmetic* as arithmetic as usual over the integers. That is, if we are working in *modulo* n, then every result x is replaced by the elements of $\{0, 1, \ldots n-1\}$ that is equivalent to x, *modulo* n. That is, x is replaced by $x \bmod n$. This informal model is sufficient if we stick to the operations of addition, subtraction and multiplication. Modular arithmetic is useful to reformulate arithmetic operations and exploit parallelism in them, where as normal methods for doing these arithmetic operations are serial in nature. A second use of modular arithmetic is in symbolic mathematical computation. This arithmetic has been found to yield efficient algorithms for the manipulations of large numbers in symbolic computations. Finally there is an intrinsic interest in this field of arithmetic by number theorists and electrical engineers specializing in communications and coding theory [CLR90, HS91].

In our problem of shortest paths and diameter in MCDL networks, the required solution is to be in integers and they have to be bounded with in the set $\{0, 1, \ldots n-1\}$. For example in equation (3.1) the number of long hops and small hops subject to (3.2) are to be in the integer range of modulo n. Equations of such nature, which require integer solutions in modulo ranges are called *linear congruences* or *diophantine equations* (after famous Alexandrian mathematician Diophantos, AD. 275).

Here we study some of the properties of modular arithmetic and solution to linear congruence equations [Ste65] as these are useful for our problem of finding shortest paths in MCDL.

B.1 Definitions

PROPERTY B.1 *The mod operator is defined as*

$$x \; mod \; y \quad = \quad x - y \cdot \lfloor x/y \rfloor \qquad if \; y \neq 0$$

$$x \; mod \; 0 \quad = \quad x \tag{B.1}$$

where x/y is fixed point integer division. ♠

From property-B.1, if x mod $y = r$, then $x = k \cdot y + r$ for integer k. The set of integers {0,1,2...p-1} where p is a prime is denoted by $GF(p)$, the Galio's field.

PROPERTY B.2 *If $(a \cdot b)$ mod $p = 1$, where p is a prime and $a, b \in GF(p)$ then a and b are said to be multiplicative inverses to each other in the modulo p domain.* ♠

The multiplicative inverse of a is represented as a^{-1} , so that $(a \cdot a^{-1})$ mod $p = 1$. So $a = b^{-1}$ is the multiplicative inverse of b in GF(p). For every element b in GF(p) except 0 there exists a unique element called b^{-1} such that $b \cdot b^{-1}$ mod $p = 1$. In order to find $a = b^{-1}$ there must exist an integer k, $0 \leq k < p$ such that $a \cdot b = k \cdot p + 1$. For example if $p = 7$, $b = 3$ and $a = 5$ then $k = 2$, so that $b^{-1} = 5$. Similarly for $b = 6$, we get $a = 6$ and $k = 5$ so that $b^{-1} = 6$. An algorithm to to compute b^{-1} is provided by the following $Euclid's$ algorithm of $greatest$ $common$ $divisor$ GCD.

ALGORITHM B.1 *$EUCLID\_ALGO(a, b)$:*
/ To find the greatest common divisor of given two integers a and b. It is assumed that $a > b \geq 0$ */*

IF $b = 0$ THEN return(a)
ELSE return EUCLID(b,a mod b))
end IF

end EUCLID. ♠

From Algorithm B.1 , it is also possible to compute two integers x, y such that $a \cdot x + b \cdot y = GCD(a, b)$. The computation of x, y is given by the $EXTENDED$ $EUCLID$ algorithm given below. But we first define a property.

PROPERTY B.3 *If $GCD(a, b) = d$ and $a, b \in GF(p)$ then there exist two integers x and y such that, $a \cdot x + b \cdot y = d$.* ♠

In Property B.3, If we let $a = p$, where p is a prime and $b \in GF(p)$, then $GCD(p, b) = 1$ and the generalisation reduces to $p \cdot x + b \cdot y = 1$. This implies that y is the multiplicative inverse of b. The following is the EX_EUCLID (extended Euclid's) algorithm to compute inverse of b, $b \in GF(p)$.

ALGORITHM B.2 *$EX\_EUCLID\_ALGO(b, p)$: /* $b \in GF(p)$, p is prime. EUCLID_ ALGO finds inverse of b. In the following the statement $(e, f) \leftarrow (g, h)$ is to be interpreted as $e \leftarrow g$; $f \leftarrow h$*/*

$(c, d, x, y) \leftarrow (p, b, 0, 1)$
while $d \neq 1$ do
$q \leftarrow c/d$

$$e \leftarrow c - d * q$$
$$w \leftarrow x - y * q$$
$$(c, d, x, y) \leftarrow (d, e, y, w)$$
repeat
if $y < 0$ *then* $y \leftarrow y + p$
return (y)

end EX_EUCLID_ALGO ♠

In the Algorithm B.2 of EX_EUCLID, the $EUCLID$ algorithm is carried in the $while$ loop [HS91]. To analyse the time for EX_EUCLID, we need to know the number of divisions in $EUCLID$ algorithm. It is given by the $Lame's$ theorem, stated below with out proof.

THEOREM B.1 *Lame's Theorem: For any integer $k \geq 1$, if $a > b \geq 0$ and $b < F_{k+1}$, where F_{k+1} is the $(k+1)th$ Fibonacci number then the invocation of $EUCLID(a, b)$ makes fewer than k recursive calls.*
proof: For proof of this refer pp 810-811 of [CLR90].

Since F_k is approximately $\frac{\phi^k}{\sqrt{5}}$, where ϕ is the golden ratio $\frac{1+\sqrt{5}}{2} = 1.61803\ldots$, the number of times the $EUCLID$ is called is $\bigcirc(log(b))$. Hence the while loop in EX_EUCLID is executed in $\bigcirc(log(b))$ time. Thus the inverse is computed in $\bigcirc(log(b))$ time.

B.2 Operators and equations

LEMMA B.1 *Additive Property:*
If a mod $m = A$ and b mod $m = B$, then $(a+b)$ mod $m = (A+B)$ mod m
Proof:

$$\begin{aligned} a \; mod \; m &= A \\ \Rightarrow a &= k_1 \cdot m + A \\ b \; mod \; m &= B \\ \Rightarrow b &= k_2 \cdot m + B \\ So,\ (a+b) &= (k_1 + k_2) + (A+B) \end{aligned}$$

Since $(k_1 + k_2)$ is an integer,

$$(a+b) \; mod \; m = (A+B) \; mod \; m$$

This proves the additive property in modular domain. ♠

LEMMA B.2 *Multiplicative Property:*
If a mod $m = A$ and b mod $m = B$ then ab mod $m = (A \cdot B)$ mod m.

Proof:

$$\begin{aligned} a \text{ mod } m &= A \qquad \Rightarrow a = k_1 \cdot m + A \\ b \text{ mod } m &= B \qquad \Rightarrow b = k_2 \cdot m + B \end{aligned}$$

So,

$$a \cdot b = k_1 \cdot m + k_2 \cdot m + (A \cdot k_1 + B \cdot k_2)\ m + A \cdot B$$

Since $(k_1 \cdot k_2 \cdot m + A \cdot k_2 + B \cdot k_1)\ m$ *is an integer, it follows that*

$$(a \cdot b) \text{ mod } m = A \cdot B$$

This proves the multiplicative property in modular domain. ♠

LEMMA B.3 *Cancelation Property:*
If $(a \cdot b)$ mod $m = a \cdot c$ and $GCD(a, m) = d$ *then* b mod $(\frac{m}{d}) = c$
Proof:

$$\begin{aligned} (a \cdot b) \text{ mod } m &= a \cdot c \\ \Rightarrow a \cdot b &= k \cdot m + a \cdot c \end{aligned}$$

$GCD(a, m) = d$ *implies that*

$$\begin{aligned} a &= a_1 \cdot d \\ m &= m_1 \cdot d \end{aligned}$$

Hence,

$$\begin{aligned} a_1 \cdot d \cdot b &= k \cdot m_1 \cdot d + a_1 \cdot d \cdot c \\ a_1 \cdot b &= k \cdot m_1 + a_1 \cdot c \end{aligned}$$

Since $GCD(a_1, m_1) = 1$, *it follows that* b *mod* $m_1 = c$, *Thus,* b *mod* $(\frac{m}{d}) = c$ ♠

The equation $(a \cdot x)$ mod $m = b$ is called the liner congruence equation. In the following we see solution for the linear congruence equations.

LEMMA B.4 $(a \cdot x)$ *mod* $m = b$ *has a unique solution for* x *if* $GCD(a, m) = 1$.
Proof: Since $GCD(a, m) = 1$, *as per property-B.3 there exist two integers* s *and* t *such that* $a \cdot s + m \cdot t = 1$. *Multiplying both sides of this equation by* b

$$\begin{aligned} a \cdot (s \cdot b) + m \cdot (t \cdot b) &= b \\ a \cdot (s \cdot b) &= -(t \cdot m)\ m + b \end{aligned}$$

Now it follows from property-B.1 that

$$(a \cdot x) \text{ mod } m = b,$$

where $x = s \cdot b$
The uniqueness of x *can be proved as below.*
Let X *be another solution to the given linear congruence equation, so that* $(a \cdot X)$ *mod* $m = b$.

Then $(a \cdot X)$ *mod* $m = (a \cdot x)$ *mod* m*. From lemma-B.3 canceling* a, X *mod* $m = x$ *mod* m*.*
Thus $X = x$ *and the solution is unique.* ♠

LEMMA B.5 $(a \cdot x)$ *mod* $m = b$ *has no solution if* $GCD(a, m) = d$ *and* d *does not divide* b*.*
Proof:

$$\begin{aligned}(a \cdot x) \text{ mod } m &= b \\ \Rightarrow a. \cdot x + m \cdot k &= b,\end{aligned}$$

where, k = *integer, from property-B.1. An integer solution for* x *and* k *in the above exists only if* a *and* m *are divisible by* b *and also* $GCD(\frac{a}{b}, \frac{m}{b}) = 1$*. In other words* $GCD(a, m) = b$*. But* $GCD(a, m) = d$ *and* d *does not divide* b*. Hence,* $(a \cdot x)$ *mod* $m = b$ *has no solution if* $GCD(a, m)$ *does not divide* b*.* ♠

LEMMA B.6 $(a \cdot x)$ *mod* $m = b$ *has exactly* d *solutions if* $GCD(a, m) = d$ *and* d *divides* b*.*
Proof:

$$\begin{aligned}(a \cdot x) \text{ mod } m &= b \\ \Rightarrow a. \cdot x + m \cdot k &= b\end{aligned}$$

where, k = *integer, from property-B.1. Since* d *divides* b, $\Rightarrow b = b_1 \cdot d$, $GCD(a, m) = d \Rightarrow a = a_1 \cdot d$ *and* $m = m_1 \cdot d$*. By substitution,* $a_1 \cdot x + m_1 \cdot k = b_1$*.*
But $GCD(a_1, m_1) = 1$*. Hence,* $(a_1 \cdot x)$ *mod* $m_1 = b_1$ *has a unique solution from property-B.3.*
The multiplicity of solutions is proved as below.
If X *is a solution then*
$X, X + m_1, X + 2m_1 + \ldots X + (d-1)m_1$ *are also the solutions as all these are modulo* m_1 *integers.* ♠

Appendix C

QUEUING THEORY AND STOCHASTIC PROCESSES

One of the most important measures of a communication network is the average delay encountered by a packet in delivering it from a source node to a destination node. The delay influences the choice of networks. Hence it is important to understand the manner in which the delay depends on the characteristics of the network, routing algorithms and flow controls. Queueing theory is the primary methodological framework for analysing the network delay. These models often provide a basis for adequate delay approximations as well as valuable qualitative results of worthwhile insights. The arrival of packets at a node in the communication network for transmitting to a destination is often studied using queueing theory. The arrival and departure processes of packets at a node is usually random. This random variable takes integral values. A powerful tool to study the integral valued random variables is by generating functions. The principle advantage of its use is that a single function (generating function) is sufficient to represent a whole set of individual items. So we also study generating functions here.

C.1 Delay components

The delay of a packet in a communication network consists of the following components [Kle75]:

- The *waiting time* (W) is the time between when the packet arrives a node for transmission and the time it starts being transmitted.
- The *transmission time* (x) is the time between the first and last bits of the packet are transmitted.
- The *propagation time* (r) is the time between a bit is transmitted at a node and the

time it is received at the next node on its route.

- The *transfer time* (T_f) is the time between the last bit is transmitted at the origin node and the time the it is received at the destination node. If we assume the average path length between a source destination pair in the network is pl_a, then $T_f = pl_a \cdot r$.

- The *service time* (T) is the time between the commencement of transmission of two successive waiting packets. This is also known as *cycle time*. Thus inverse of service time gives rate of service ($\mu = \frac{1}{T}$).

The above accounting neglects retransmission due to transmission errors, processing time required for a packet in assigning to the transmit buffer in the node and processing time required for assembling the received packet.

C.2 Queueing models

We consider queueing system where packets arrive at *random* to obtain *service*. We are interested in estimating quantities such as, the average delay for a packet. The *average delay* of a packet represented by D_a, is obtained from the delay components defined in Section C.1 as $D_a = W + x + T_f$. This quantity will be estimated in terms of known information such as:

- The *packet arrival rate i.e.*, the typical number of packets entering the system per unit time. Often we use the term *interarrival time* defined as the time between two successive arrivals to characterise the arrival rate.

- The *packet service rate i.e.*, the typical number of packets that the system serves per unit time (μ).

- The available *buffer space* for the packets to store (wait) while one is already in service point and receiving service.

- The *number of servers* which do the service (transmission) for the packets that have arrived in the buffer.

The interarrival and service times may be *deterministic* or *chance-dependent* also called *random*. If they are deterministic the case is trivial. We are concerned with the case of chance-dependent *stochastic* behavior of interarrival and service times. The mean arrival rate and service rate are generally represented by λ and μ. The ratio $\rho = \frac{\lambda}{\mu}$ is called *traffic intensity*. A convenient notation to denote queueing model is by a three part descriptor, $A/B/C$, where A = interarrival time distribution, B = service time distribution and C =number of servers.

Table C.1: Some probability distributions, their mean and variances

Distribution	$p.g.f$	Mean	Variance
Poisson (λ)	$e^{\lambda(s-1)}$	λ	λ
Binomial (M, p)	$(p \cdot s + q)^M$	$M \cdot p$	$M \cdot p \cdot q$
Geometric (p)	$\frac{p}{1-q \cdot s}$	$\frac{q}{p}$	$\frac{q}{p^2}$

The A and B usually take the symbols M for exponential distribution, E_k for Erlang k-distribution, G for arbitrary general distribution and D for fixed (deterministic) arrival. For example $M/M/1$, $M/G/1$, $G/G/m$ are some queueing disciplines. In the following we describe these distributions briefly.

C.3 Stochastic processes

In a physical process if one of its parameters say, X assumes values randomly then we call such a variable as *stochastic variable* and call the process as *stochastic process*. Suppose that X assumes non-negative random values $0, 1, 2 \ldots$ and that $Pr[X = k] = p_k$ then $\Sigma_k \; p_k = 1$. Now let s be another variable. We define a function $P(s) = \Sigma_{k=0}^{\infty} \; p_k \cdot s^k$. The $P(s)$ is called probability generating function, $p.g.f$ of the sequence of probabilities p_k of the random variable X. The mean (first), second moment and variance of random variable X can be obtained from the $p.g.f$ of X as [Med94]:

Mean or first moment of X is

$$E[X] = [\frac{d}{ds} \cdot P(s)]_{(s=1)} \tag{C.1}$$

Second moment of X is

$$E[X^2] = [\frac{d^2}{ds^2} \cdot P(s)]_{(s=1)} + [\frac{d}{ds} \cdot P(s)]_{(s=1)} \tag{C.2}$$

Variance of X is

$$\sigma_X^2 = E[X^2] - (E[X])^2 \tag{C.3}$$

In Table C.1 we give some important distribution functions, their $p.g.f$, mean and variances.

THEOREM C.1 *The $p.g.f$ of the sum of two independent random variables X and Y is the product of the $p.g.f$ of X and Y.*

Proof: Refer [Med94] for the proof. ♠

Hence if X, Y and Z are random variables and $Z = X + Y$ with their $p.g.f$ as $X(s)$, $Y(s)$ and $Z(s)$ then $Z(s) = X(s) \cdot Y(s)$.

THEOREM C.2 *If the random variable X has the $p.g.f$ $X(s)$ then $a \cdot X + b$, where a and b are constants has the $p.g.f$ $s^b \cdot X(s^a)$*

Proof: Refer [Med94] for proof. ♠

C.4 Little's result

One of the important results often used in Queueing models is $Little's$ result. It states that average number of customers (packets) in the queue is equal to the average arrival rate of customers to the system, times the average waiting time in the queue. For proof refer [Kle75]. Thus,

$$Q = \lambda \cdot W \tag{C.4}$$

where $Q =$ average number of packets in node buffer, $\lambda =$ average arrival rate and $W =$ average waiting time.

C.5 $M/G/1$ results

In what follows we present an understanding of $M/G/1$ model and the results there in [Kle75]. The $M/G/1$ Queueing system is a single server system with Poisson arrival and arbitrary service time distribution. That is, the interarrival time distribution is exponentially distributed with an average l rate of λ arrrivals per second. So the mean interarrival time is $\frac{1}{\lambda}$ sec, and variance of arrivals is λ. Let the server is available at random time intervals T, which is *generally distrbuted* and serves only one of the arrivals (*non-exhaustive service*). Let $\overline{T}$, $\overline{T^2}$ and σ_{T^2} denote mean, second moment and variance of T respectively. We can also say that T is the time for server revisits. We then have *traffic intensity*:

$$\rho = \lambda \cdot \overline{T} \tag{C.5}$$

The ρ is also an indication of the probability that at least one customer is in the queue for service or queue length is at least one. The queue is stable if $\rho < 1$. So we shall assume that $\rho < 1$. Then the waiting time W and its mean $\overline{W}$ are obtained as follows:

Let Q be the length of the queue of arrivals and $\overline{Q}$ its mean. the server. Note that even if $Q = 0$, it may or may not find the server as the server reviits are random. If $Q > 0$, it waits the queue ahead of it is serviced ($Q = 0$) and the server is available to it. So let us define $T_R$as the time from occurance of an arival at the head of the queue to the time server is available. T_R is called the *residual life* of server $\overline{T_R}$ is its mean. Obviously T_R is also a random varible. So we have the waiting time W of an arrival

$$W = T_r + T \cdot Q$$
$$\overline{W} = \overline{T_r} + \overline{T} \cdot \overline{Q}$$

Using (C.4)

$$\begin{aligned}
\overline{W} &= \overline{T_r} + \overline{T} \cdot \lambda \overline{W} \\
\overline{W} &= \frac{\overline{T_r}}{1 - \lambda \overline{T}} \\
&= \frac{\overline{T_r}}{1 - \rho}
\end{aligned}$$

The mean residual life $\overline{T_r}$ is given by *Pollaczek-Khinchin* formula, P-K mean value formula [Kle75]

$$\begin{aligned}
\overline{T_r} &= \frac{\overline{T^2}}{2\overline{T}} \\
&= \frac{\sigma_T^2 + \overline{T}^2}{2\overline{T}} \\
&= \frac{(1 + C_b^2)\overline{T}}{2}
\end{aligned}$$

where $C_b^2 = \frac{\sigma_T^2}{\overline{T}^2}$ is called coeffiecient of varaition of T. Thus we have the average waititng time $\overline{W}$ given by

$$\begin{aligned}
\overline{W} &= \frac{(1 + C_b^2)\overline{T}}{2(1 - \rho)} \\
&= \frac{(1 + C_b^2)\rho}{2\lambda(1 - \rho)} \qquad \text{(C.6)}
\end{aligned}$$

Bibliography

[AK91] B. W. Aeysundara and A. E. Kamal. High Speed Local Area Networks and their Performance. *ACM Comput. Surv.*, 23(2):221–263, 1991.

[AL81] B.W. Arden and H. Lee. Analysis of Chordal Ring Network. *IEEE Trans. on Computers*, C-30(4):291–295, 1981.

[BBC95] V. Bala, J. Bruck, and R. Cypher. CCL: A Portable and Tunable Collective Communication Library for Scalable Parallel Computers. *IEEE Trans. on Parallel and Dist. Systems*, 6(2):154–164, 1995.

[BC96] Rajendra V. Boppana and Suresh Chalasani. A Framework for Designing Dealock-free Wormhloe Routing Algorithms. *IEEE Trans. on Parallel and Dist. Systems*, 7(2):169–183, 1996.

[BCH95] J.C. Bermond, F. Comellas, and D.F. Hsu. Distributed Loop Computer Networks : A Survey. *Journal of Parallel and Distributed Computing*, 24:2–10, 1995.

[BGY89] Laxmi N. Bhuyan, D. Ghosal, and Q. Yang. Approximate Analysis of Single and Multiple Ring Networks. *IEEE Trans. on Computers*, C-38(7):1027–1040, 1989.

[BT91] J. C. Bermond and D. Tzvieli. Minimal Diameter Double-Loop Networks : Dense Optimal Families. *Networks*, 21:1–9, 1991.

[Bux81] W. Bux. Local-Area Subnetworks:A Performance Comparison. *IEEE Trans. on Commmunication*, COM-29(10):1465–1473, 1981.

[Bux84] W. Bux. Performance Issues in Local-area Networks. *IBM Systems Journal*, 23(4):351–374, 1984.

[BW85] F.T. Boesch and J.F. Wang. Reliable Circulant Networks with Minimum Transmission Delay. *IEEE Trans on Circuits and Systems*, CAS-32(12):1286–1291, 1985.

[Car79] B. Carre. *Graphs and Networks*, pages 50–51. Oxford, England, 1979.

[CB97] N. Chalamaiah and R. Badrinath. Communication Primitives in Ring Connected Multicomputers. In *Proceeding of IEEE National Conference on INTERNET for INDIA, Hyderabad, India*, pages 17–20. IEEE Hyderabad chapter, 11-13,Dec 1997.

[CB98a] N. Chalamaiah and R. Badrinath. A Multitoken Distributed Loop for Local Area Networks. In *Proceedings of 21 st National Systems Conference on Integrated Systems with Multiple Technologies NSC-97*, pages 333–338. Research Center,IMARAT,DRDO,Hyderabad, Allied Publishers, 22-24, Jan 1998.

[CB98b] N. Chalamaiah and R. Badrinath. Analysis of Multiconnected Loop Topologies for Interconnection Networks. In *Proceedings of 21 st National Systems Conference on Integrated Systems with Multiple Technologies,NSC-97*, pages 327–332. Research Center,IMARAT,DRDO,Hyderabad, Allied Publishers, 22-24, Jan 1998.

[CB98c] N. Chalamaiah and R. Badrinath. Finding Shortest Paths in Distributed Loop Networks. *Information Processing Letters*, 67(3):157–161, 1998.

[CB98d] N. Chalamaiah and R. Badrinath. Modelling and Analysis of Multitoken Multiple Loop Networks. In *Proceedings of International Conference on Networking India and the World,CNIW-98*, pages 49–55. IEEE Gujarat Section and IEEE India Council, 7-12,Dec 1998.

[CB98e] N. Chalamaiah and R. Badrinath. Multiconnected Distributed Loop Interconnection Networks. In *Proceedings of International Conference on Information Technology,CIT-98*, pages 181–184. Orissa Information Technology Society & Utkal University, Bhubaneswar,India, TMH Publishers, 21-23,Dec 1998.

[CB98f] N. Chalamaiah and R. Badrinath. Multiple Token Distributed Loop Local Area Networks: Analysis. In *Proceedings of 5th International Conference on High Performance Computing,HiPC98,Chennai,India*, pages 400–407. IEEE Computer Society Technical Committee on Parallel Processing, IEEE Computer Society Press, 17-20,Dec 1998.

[CB99a] N. Chalamaiah and R. Badrinath. Analysis Modelling and Simulation of Multitoken Multiconnected Distributed Loop Local Area Networks. In *Proceedings of the 5th National Conference on Communications NCC-99*, pages 417–425. Indian Institute of Technology,Kharagpur,India, Allied Publishers, 29-31,Jan 1999.

[CB99b] N. Chalamaiah and R. Badrinath. Analysis Modelling and Simulation of Multitoken Multiconnected Distributed Loop Local Area Networks. *Computer Communications*, 1999. communicated.

[CBS98] N. Chalamaiah, R. Badrinath, and R.Lalith Shankar. Modelling and Analysis of Multitoken Multiconnected Loop Networks. In *Proceedings of International conference on Computer and Information Technology*, pages 2–7. Bangldesh University of Engineering and Technology, Dhaka,Bangladesh, 18-20,Dec 1998.

[CLL82] R. Churukuri, L.Li, and L.Louis. Evaluation of Token Passing Schemes in Local Area Networks. *Proc. of IEEE Computer Networking Symposium*, pages 57–68, 1982.

[CLR90] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to algorithms*, chapter Number Theoretic Algorithms, pages 801–844. The MIT Press, 1990. Reading.

[CO93] Israel Cidon and Yorm Ofek. MetaRing– A Full-Duplex Ring with Fairness and Spatial Reuse. *IEEE Trans. on Commmunication*, 41(1):110–120, 1993.

[CPR78] David D. Clark, Kenneth T. Pogran, and David P. Reed. An Introduction to Local Area Networks. *Proc. of IEEE*, 66(11):1497–1517, 1978.

[DA93] William J. Dally and Hiromichi Aoki. Deadlock-free adaptive routing in multicomputer networks using virtual channels. *IEEE Trans. on Parallel and Dist. Systems*, 4(4):466–475, 1993.

[Dal92] William J. Dally. Virtual Channle Flow Control. *IEEE Trans. on Parallel and Dist. Systems*, 3(2):194–205, 1992.

[Deo97] Narsingh Deo. *System simulation with Digital Computer*, chapter Discrete System Simulation, pages 40–61. Prentice Hall India Pvt Ltd, 1997. Reading.

[DH90] D.Z. Du and D.F. Hsu. A Combinatorial Problem Related to Distributed Loop Networks. *Networks*, 20:173–180, 1990.

[DS87] W.J. Dally and C.L. Seitz. Deadlock-free Message Routing in Multiprocessor Interconnection Networks. *IEEE Trans. on Computers*, C-36:547–553, May,1987.

[DYN97] Jose Duato, Sudhakar Yelamanchili, and Lionel Ni. *Inerconnection Networks: an Engineering Approach*, chapter 2 to 4 and 9, pages 35–168 and 401–472. IEEE Computer Society, 1997.

[EAF93] P. Esque, F. Aguilo, and M.A. Fiol. Double Commutative Step Digraph with Minimum Diameter. *Discrete Math.*, 114:147–157, 1993.

[EH92] Paul Erdoos and D.Frank Hsu. Distributed Loop Network with Minimum Transmission Delay. *Theoretical Computer Science*, 100:223–241, 1992.

[Fen81] T.Y. Feng. A Survey of Inteconnection Networks. *IEEE Computer*, pages 12–27, December, 1981.

[FL72] D. J. Farber and K. C. Larson. The Structure of a Distributed Computer System Software. *Symp. on Computer Communication Networks and Teletraffic*, Polytechnic Institute of Brooklyn:539–545, Apr,1972.

[FN69] W. D. Farmer and E. E. Newhall. An Experimental Distributed Switching System to Handle Bursty Computer Traffic. *Proc. ACM Symposium on Problems in the Optimisation of Data Communication*, pages 1–33, Oct,1969.

[FYAV87] Miguel A. Fiol, J. Luis A. Yebra, Ignacio Alegre, and Mateo Valero. A Discrete Optimisation Problem in Local Networks and Data Alignment. *IEEE Trans. on Computers*, C-36(6):702–713, 1987.

[GKG80] A. Grnanov, L. Klienrock, and M. Gerla. A Highly Reliable Distributed Loop Networks Architecture. *Proc.of Int'l symp. Fault-Tolerant Computing, Japan*, pages 319–324, 1980.

[HB89] Kai Hwang and Faye A. Briggs. *Computer Architecture and Parallel Processing.* Mc Graw-Hill. Inc., 1989.

[HH92] X.D. Hu and F.K. Hwang. Reliabilities of Chordal Rings. *Networks*, 22:487–501, 1992.

[HK94] Leendert M. Huisman and Sandip Kundu. Highly Reliable Symmetric Networks. *IEEE Trans. on Parallel and Dist. Systems*, 5:94–97, 1994.

[HS91] E. Horowitz and S. Sahani. *Fundamentals of Computer Algorithms*, chapter Algebraic Simplification and Transformation, pages 440–447. Galgotia Publications, 1991.

[HW95] F.K. Hwang and P.W. Wright. Survival Reliability of Some Double-Loop Networks and Chordal Rings. *IEEE Trans. on Computers*, C-44(12):1468–1471, 1995.

[Hwa93] Kai Hwang. *Advanced Computer Architecture: Parallelism, Scalability, Programmability.* Mc Graw-Hill. Inc., 1993.

[IEE82] IEEE. *Draft Standard 802.4: Token Passing Bus Access Methods and Physical Layer Specifications: Draft D*, December 1982.

[Kam94] Ahmed E. Kamal. The Multi-Token Ring Network Protocol. *Comput. Networks and ISDN Systems*, 26:1477–1494, 1994.

[Kle75] L. Kleinrock. *Queueing Systems: Vol 1*, chapter : The Queue G/G/1, pages 275–318. John Wiley Sons, 1975.

[KM72] A.G. Konheim and B. Meister. Studies in a Loop System. *J.ACM*, 19(1):92–108, 1972.

[Kuc79] P.J. Kuchen. Multiqueue Systems with Non-exhaustive Cyclic Services. *Bell Systems Technical Journal*, 58:671–698, 1979.

[LH91] Danie H. Linder and Jim C. Harden. An Adaptive and Fault Tolerant Wormhole Routing Strategy for k-ary-n cubes. *IEEE Trans. on Computers*, 40(1):1–12, 1991.

[Liu78] M.T. Liu. Distributed Loop Computer Networks. *Advances in Computers*, 17:163–221, 1978.

[Med94] J. Medhi. *Stochastic Processes, 2nd edition*, chapter : Stochastic Processes in Queueing and Reliability, pages 407–523. Wiley Eastern Limited, 1994.

[MFMK93] R. Mukkamala, E. C. Foudriat, K. J. Maly, and V. Kale. Modelling and Analysis of High Speed Parallel Token Rings. *Proc. of 17th IEEE Int'l Conference on Local Computer Networks*, pages 624–633, 1993.

[MI89] H. Masuyama and T. Ichimori. Tolerance of Double-Loop Computer Networks to Multinode Failures. *IEEE Trans. on Computers*, C-38(5):738–741, 1989.

[MS95] K. Mukhopadhyaya and B.P. Sinha. Fault-Tolerant Routing in Distributed Loop Networks. *IEEE Trans. on Computers*, C-44(12):1452–1456, 1995.

[NM93] Lionel M. Ni and Philip K. McKinley. A Survey of Wormhole Routing Techniques in Direct Networks. *IEEE Computer*, 26(2):62–76, 1993.

[Par95] Behrooz Parhami. Periodically Regular Chordal Ring Networks for Massively Parallel Architectures. *5th Symp. Frontiers of Massively Parallel Computation*, pages 315–322, Feb, 1995.

[PB79] B. K. Penny and A.A. Baghadadi. Survey of Computer Communication Loop Networks: Parts 1 and 2. *Computer Communications*, 2:165–180 and 244–251, 1979.

[Pie72] J.R. Pierce. Networks for Block Switching of Data. *Bell Systems Technical Journal*, 51(6):1133–1145, 1972.

[PR82] D. K. Pradhan and S. M. Reddy. A Fault-Tolerant Communication Architecture for Distributed System. *IEEE Trans. on Computers*, C-31(9):863–869, 1982.

[PT92] J.M. Peha and F.A. Tobagi. Comments on Tolerance of Double-Loop Computer Networks to Multi Node Failure. *IEEE Trans. on Computers*, C-41(11):1488–1489, 1992.

[QLL92] Yaoshuang Qu, Lawrence H. Landweber, and Miron Livny. Parallelring: A Token Ring LAN with Concurrent Multiple Transmissions and Message Destination Removal. *IEEE Trans. on Commmunication*, 40(4):738–745, 1992.

[Ren80] D.A. Rennels. Distributed Fault-tolerant Computer Systems. *IEEE Computer*, 13:55–66, May 1980.

[RGA85] C.S. Raghavendra, M. Gerla, and A. Avizienis. Reliable Loop Topologies for Large Local Computer Networks. *IEEE Trans. on Computers*, C-34(1):46–55, 1985.

[RS85] C.S. Raghavendra and J.A. Silvester. Double Loop Network Architectures A Performance Study. *IEEE Trans. on Computers*, COM-33(2):185–187, 1985.

[RS86] C.S. Raghavendra and J.A. Silvester. A Survey of Multi-Connected Loop Topologies for Local Computer Networks. *Comput. Networks and ISDN Systems*, 11:29–42, 1986.

[Sha99] R. Lalith Shankar. Simulation of Multiconnected Multitoken Ring Networks. Master's thesis, Indian Institute of Technology, Department of Computer Science and Engineering, Kharagpur, India, 1999.

[SLCH98] Ting-Yi Sung, Chun-Yuan Lin, Yen-Chu Chuang, and Lih-Hsing Hsu. Fault Tolerant Token Ring Embedding in Double Loop Networks. *Information Processing Letters*, 66:201–207, 1998.

[SS85] Adarshpal S. Sethi and T. Saydam. Performance Analysis of Token Ring Local Area Networks. *Comput. Networks and ISDN Systems*, 9:191–200, 1985.

[Ste65] B. M. Stewart. *Theory of Numbers*, chapter Chapters 7,14 and 15. The Macmillan Company Publication, 1965.

[Tzv91] D. Tzvieli. Minimal Diameter Double-Loop Networks.I.Large Infinite Families. *Networks*, 21:387–415, 1991.

[vA94] Harmen R. van As. Media Access Techniques: The Evolution Towards Terabit/s LANs and MANs. *Comput. Networks and ISDN Systems*, 26:603–656, 1994.

[VR94] D. A. Vasko and R. Rajsuman. Analysis and Simulation of Multiple-Ring Token Networks. *IEE proc.-Comput.Digit.Tech*, 141(2):135–141, 1994.

[WC74] C.K. Wong and D. Coppersmith. A Combinatorial Problem Related to Multimode Memory Organisation. *J.ACM*, 21:392–402, 1974.

[Wil72] Robert S. Wilkov. Analysis and Design of Reliable Computer Networks. *IEEE Trans. on Commmunication*, COM-20(3):660–678, june,1972.

[WK93] May-Yew Wee and Ahmad E. Kamal. A Partial Destination Release Strategy for Multi-Token Ring Protocol. *Proc. of 17th IEEE Int'l Conference on Local Computer Networks*, pages 639–638, 1993.

[ZP93] J. Zerovnik and T. Pisanski. Computing Diameter in Multiple-Loop Networks. *Journal of Algorithms*, 14:226–243, 1993.

Contents

List of Figures

List of Tables