National Library of Canada
Acquisitions and Bibliographic Services Branch
395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale du Canada
Direction des acquisitions et des services bibliographiques
395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Canada

A Software Based Design Space Exploration of a Free-Space Photonic Backplane

Manoj Verghese,
(B.Eng 1992)

McGill University, Montreal

June 1995

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

ISBN 0-612-07988-0

Canada

Abstract

Optics is receiving increasing attention as an attractive technology to implement high performance backplanes. The growth in the fields of digital telecommunications and parallel computing is demanding backplanes with ever-increasing interconnection densities and data transfer rates [2, 5].

As a possible solution to the limitations of electrical backplanes [9], research into free-space photonic backplanes is underway at McGill University [1]. The field of research into photonic backplane architectures is new and relatively small amounts of literature are available on the subject. Hence, the research effort at McGill represents one of the first architectural level studies in the field of free-space photonic backplanes.

As part of the research program underway at McGill, a photonic backplane architecture has been proposed [27] and is currently under construction [1]. Due to the novelty of the backplane architecture there was a need for a preliminary investigation into its feasibility and capabilities. A large analytic performance model which could analyze the numerous architectural variations of the backplane has been developed [25]. In this thesis a software tool, based on the analytic model in [25], has been developed for broad design space explorations of the backplane architecture. The modelling approach is general and can be extended to the analysis of arbitrary networks and backplane architectures. However, this thesis will focus on the design space exploration of internally nonblocking ATM switches on various backplane architectures. Throughout the thesis, the synchrononous smart pixel array design from June 1994 will be assumed.

Résumé

De plus en plus, la technologie optique se dévoile comme étant une solution viable aux systèmes d'interconnexions à haute performance des cartes électroniques. Les avancements dans les domaines des télécommunications digitales et de l'informatique des ordinateurs parallèles requièrent des systèmes d'interconnexions à densités grandissantes et des transferts de données à débits croissants [2, 5].

Les recherches entreprises à l'université McGill [1] sur les systèmes d'interconnexions optiques offrent une solution potentielle aux limitations technologiques des systèmes d'interconnexions électriques [9]. Le domaine de recherche axé sur l'architecture des systèmes d'interconnexions optiques est récent et jusqu'à présent très peu de travaux ont été publiés. Les recherches à McGill représentent donc un des premiers efforts dans le domaine de l'architecture des systèmes d'interconnexions optiques.

Un des éléments de la recherche à McGill consiste à la conception [27] et la mise en œuvre [1] d'une architecture pour un système d'interconnexions optiques. *A priori* une étude de faisabilité était nécessaire étant donné l'aspect pionnier d'un tel projet. Un modèle analytique complexe a été developpé [25] pour permettre une évaluation de la performance sous divers paramètres architecturaux. Cette thèse présente un logiciel basé sur le modèle analytique de [25]. Le logiciel fournit un outil d'exploration à l'ensemble des concepts possibles pour un système d'interconnexions optiques. Le modèle utilisé se veut général de sorte à être applicable à l'analyse de réseaux arbitraires. Cependent, cette thèse porte plus particulièrement sur l'exploration du design de commutateurs ATM à congestion interne basés sur fonds de paniers variés. Tout au long de cette thèse, le design du smart pixel array synchrone datant de juin 1994 sera sous-entendu.

Acknowledgements

The author wishes to gratefully acknowledge the thesis supervision of Professor Ted Szymanski. His enthusiasm and encouragement have made this thesis possible. The opportunity to work in this new and exciting field of research was most appreciated.

I wish to thank to Jacek Slaboszewicz, whose unwavering dedication to the Microelectronics and Computer Systems laboratory kept the computer systems running smoothly, day or night. He was always willing to answer my many questions whenever I ran into difficulties.

Thanks to my fellow students in the lab, especially Eric, with whom I could always count on to engage in stimulating and enlightening discussions on a myriad of topics. Thanks to Betina who gave me a warm welcome to the lab and having shown me its social side when I started in the fall of 1992. The friendships I formed made my stay in the lab an enjoyable one.

Thanks to my dear friend Kris for her companionship over the last three years. The conversations we shared always made the daily commute to and from the university a pleasure.

Finally, thanks to my parents, for it is with their constant support and guidance that I chose to pursue my studies into graduate school.

Funding Acknowledgments

The Canadian Institute for Telecommunications Research (CITR), a member of the Center of Excellence program in Canada, is funding a Major Project in Photonic Devices and Systems. The Major Project is organized into four projects located throughout Canada, including the "Optoelectronic Devices", "Optoelectronic Packaging Concepts", "Optical and Optomechanical Hardware" and "Large ATM Architectures" projects [1].

One goal of the Major Project is to demonstrate a free-space photonic backplane based on smart pixel arrays. To meet this objective, a photonic backplane architecture has been proposed and a mathematical performance model has been developed [25]. The backplane architecture is currently under development at McGill University and a demonstrator which interconnects four printed circuit boards in planned for the Fall of 1995.

In this thesis a software tool which implements the mathematical performance model in [25] has been developed. The research contained in this thesis was supported in part by NSERC Canada Grant OGP0121601 and by the CITR through Project 94-3-4 entitled "Large ATM Architectures". This research was performed in the Microelectronics and Computing Systems Laboratory (i.e., the MACS Lab.) at McGill University. The use of computing equipment on loan to the MACS Laboratory from the Canadian Microelectronics Corporation (CMC) through its 1993, 1994 and 1995 Equipment Loans is also acknowledged.

Contents

Abstract i

Résumé ii

Acknowledgements iii

1 Introduction 1

1.1 Motivation 2

1.1.1 Electrical Backplanes 2

1.1.2 Free-Space Optical Backplanes 3

1.2 Author's Contribution 5

1.3 Overview 6

2 Overview of the Hyperplane 8

2.1 Structure 8

2.1.1 Parallel Optical Channels 8

2.1.2 Nodes 9

2.1.3 Smart Pixel Arrays 10

2.1.4 Mode of Operation 12

2.2 Alternate Structures of the Basic Hyperplane 13

2.2.1 Single-Stream Circular Hyperplane 13

2.2.2 Dual-Stream Circular Hyperplane 13

2.3 Embeddings 14
2.3.1 Linear Hyperplane 16
2.3.2 Dual-Stream Circular Hyperplane 22
2.4 Summary 27

3 Design Goals of the Software Tool **28**
3.1 Hardware Configurations 29
3.2 Operating Conditions 30
3.3 Performance Measures 31
3.3.1 Unbuffered Analysis 31
3.3.2 Buffered Analysis 32
3.4 Modeling Accuracy and Computation Complexity 32
3.4.1 Unbuffered Analysis 32
3.4.2 Buffered Analysis 33
3.5 Software Interface 33
3.5.1 Numerical Analysis Interface 34
3.5.2 Numerical Visualization Interface 35
3.6 Hierarchical Design 36
3.6.1 Numerical Analysis Software Structure 36
3.6.2 Numerical Visualization Software Structure 37
3.7 Summary 38

4 Analytic Model of the Hyperplane **39**
4.1 Theoretical Performance Analysis 39
4.2 Packet Time-Slot Duration 40
4.2.1 Embeddings in the Linear Hyperplane 40
4.2.2 Embeddings in the Dual-Stream Circular Hyperplane 41

4.3 Probability of Packet Blocking and Acceptance 42
4.3.1 Single-Stream Circular Hyperplane 42
4.3.2 Linear Hyperplane - Sequential Channel Assignment 44
4.3.3 Linear Hyperplane - Interleaved Channel Assignment 47
4.3.4 Dual Stream Circular Hyperplane 52
4.3.5 Poisson Model of the Binomial 55
4.4 Network Throughput 57
4.4.1 Linear Hyperplane 57
4.4.2 Dual-Stream Circular Hyperplane 59
4.5 Input Queuing Analysis 59
4.5.1 $M \mid M \mid Y \mid \infty$ Queue with Y Servers 60
4.5.2 $M \mid M \mid Y \mid Q_s$ Queue with Y Servers 62
4.6 Internally Nonblocking Networks 64
4.6.1 Crossbar 64
4.6.2 Knockout Switch 64
4.6.3 Fully Connected Network 64
4.6.4 Dilated Crossbar 65
4.6.5 Crossout Switch 65
4.7 Summary 65

5 Software Operation **66**

5.1 Introduction 66
5.2 Numerical Analysis 66
5.2.1 Embedded Networks 72
5.3 Numerical Visualization 74
5.4 Analysis Considerations 79
5.5 System Requirements 80

6 Numerical Results 81

6.1 Introduction 81

6.2 Linear Hyperplane 82

6.2.1 Unbuffered Analysis 82

6.2.2 Buffered Analysis 85

6.3 Dual-Stream Circular Hyperplane 87

6.3.1 Maximized Edge Bandwidth Embedding 87

6.3.2 Minimized Propagation Delay Embedding 89

6.3.3 Optimized Edge Bandwidth and Propagation Delay Embedding . . . 89

6.4 Effect of Hardware Parameters and Operating Conditions 90

6.4.1 Effect of the Network Size on Performance 90

6.4.2 Effect of the Offered Load on Performance 91

6.4.3 Effect of the Number of Optical Bit-Channels on Performance 91

6.4.4 Effect of the Clock Frequency on Performance 94

6.4.5 Effect of the Packet Size on Performance 95

6.5 Summary 97

7 Conclusions 99

Bibliography 101

A Numerical Analysis Software 104

A.1 Top Level Routine for the Dilated Crossout Switch 104

A.2 Probability of Blocking Routine for the Dilated Crossout Switch 107

List of Figures

1.1 Board-to-board interconnect with bulk optics 3

1.2 Board-to-board interconnect with micro-optics 4

1.3 Board-to-board interconnect with holographic elements 5

2.1 Free-space photonic backplane 9

2.2 A basic smart pixel array 10

2.3 Variations of the basic smart pixel array 11

2.4 Single-stream circular hyperplane 13

2.5 Dual-stream circular hyperplane 14

2.6 Embedding template of the hyperplane 15

2.7 An $N = 16$ Crossbar switch 17

2.8 Hypergraph model of an $N = 16, L = 8$ Knockout switch 18

2.9 An $N = 16, a = 2, b = 4$ dilated Crossbar switch 19

2.10 An $N = 16$ Fully Connected network 21

2.11 An $N = 16, C = 4, K = 4$ Crossout switch 23

2.12 An $N = 16, C = 8, K = 4, a = 2, b = 4$ dilated Crossout switch 24

2.13 Maximized edge bandwidth embedding of a Crossout switch 25

2.14 Minimized propagation delay embedding of a Crossout switch 26

2.15 Optimized bandwidth and propagation delay embedding of a Crossout switch 27

3.1 A window of the numerical analysis interface 34

3.2 A window of the numerical visualization interface 35
3.3 Structure of numerical analysis section of the software tool 36
3.4 Structure of the numerical visualization section of software tool 37

4.1 Embedding of an $N = 16, C = 4, K = 4$ Crossout in the single-stream circular hyperplane 43
4.2 Sequential assignment of nodes to logical channels 45
4.3 Interleaved assignment of nodes to logical channels 48
4.4 Blocking probability for interleaved versus sequential channel assignment . . 51
4.5 Maximized edge bandwidth embedding of the $N = 16, K = 4, C = 4$ Crossout switch in the dual-stream circular hyperplane 53
4.6 Minimized propagation delay embedding of the $N = 16, K = 4, C = 4$ Crossout switch in the dual-stream circular hyperplane 54
4.7 Maximized edge bandwidth and minimized propagation delay embedding of the $N = 16, K = 4, C = 4$ Crossout switch in the dual-stream circular hyperplane 55
4.8 Markov chain model for an $M \mid M \mid Y \mid \infty$ queue 60
4.9 Markov chain model for an $M \mid M \mid Y \mid Q_s$ queue 62

5.1 Hyperplane Architecture selection menu 67
5.2 Channel Assignment selection menu 68
5.3 Embedding Scheme selection menu 68
5.4 Probability Computation Model selection menu 69
5.5 Hardware Settings and Operating Conditions window 70
5.6 Parameters of the Design Space Exploration window 75
5.7 Network Size and Offered Load Point window 76
5.8 Curve Menu 77
5.9 An output graph window 78

3.2 A window of the numerical visualization interface 35
3.3 Structure of numerical analysis section of the software tool 36
3.4 Structure of the numerical visualization section of software tool 37

4.1 Embedding of an $N = 16, C = 4, K = 4$ Crossout in the single-stream circular hyperplane 43
4.2 Sequential assignment of nodes to logical channels 45
4.3 Interleaved assignment of nodes to logical channels 48
4.4 Blocking probability for interleaved versus sequential channel assignment . . 51
4.5 Maximized edge bandwidth embedding of the $N = 16, K = 4, C = 4$ Crossout switch in the dual-stream circular hyperplane 53
4.6 Minimized propagation delay embedding of the $N = 16, K = 4, C = 4$ Crossout switch in the dual-stream circular hyperplane 54
4.7 Maximized edge bandwidth and minimized propagation delay embedding of the $N = 16, K = 4, C = 4$ Crossout switch in the dual-stream circular hyperplane 55
4.8 Markov chain model for an $M \mid M \mid Y \mid \infty$ queue 60
4.9 Markov chain model for an $M \mid M \mid Y \mid Q_s$ queue 62

5.1 Hyperplane Architecture selection menu 67
5.2 Channel Assignment selection menu 68
5.3 Embedding Scheme selection menu 68
5.4 Probability Computation Model selection menu 69
5.5 Hardware Settings and Operating Conditions window 70
5.6 Parameters of the Design Space Exploration window 75
5.7 Network Size and Offered Load Point window 76
5.8 Curve Menu 77
5.9 An output graph window 78

5.10 Three dimensional analysis space of the numerical analysis routine 79

6.1 Network throughput in the linear hyperplane using the June 1994 smart pixel array design 82

6.2 Losses in the linear hyperplane using the June 1994 smart pixel array design 83

6.3 Packet time-slot duration characteristics in the linear hyperplane using the June 1994 smart pixel array design 84

6.4 Buffered analysis of the linear hyperplane using the June 1994 smart pixel array design 86

6.5 Losses for buffered linear hyperplane using the June 1994 smart pixel array design 87

6.6 Unbuffered analysis of the dual-stream circular embeddings using the June 1994 smart pixel array design 88

6.7 Effect of offered load on bandwidth and blocking probability using the June 1994 smart pixel array design 92

6.8 Effect of Z (optical bit-channels) on the packet time-slot duration using the June 1994 smart pixel array design 93

6.9 Effect of Z (optical bit-channels) on the aggregate bandwidth using the June 1994 smart pixel array design 94

6.10 Effect of B (clock frequency) on the aggregate bandwidth using the June 1994 smart pixel array design 95

6.11 Effect of P (packet size) on the packet time-slot duration using the June 1994 smart pixel array design 96

6.12 Effect of P (packet size) on the aggregate bandwidth using the June 1994 smart pixel array design 97

Chapter 1

Introduction

A new research program, the Photonic Systems and Devices Major Project, has been undertaken at McGill University under the auspices of CITR, the Canadian Institute for Telecommunications Research. The goal of this Major Project is to construct a free-space optical backplane with a terabit/s aggregate bandwidth. Concurrent with its development, it is desired to identify novel, large scale ATM switching architectures made possible by the high connectivity and throughput envisioned for the backplane. Such a backplane would have applications for large scale systems in the fields of information processing, telecommunications and parallel computing [1].

Over more than one year's time, a preliminary conceptual model of the free-space photonic backplane gradually took shape. During its evolution, the need arose for analytic material to model the backplane. Due to the extremely large number of different architectural configurations and operating conditions for the backplane, it became vital to have a software tool which could implement the analytic material.

The goal of the work described herein was to produce this software tool for the purpose of performing an analysis or design space exploration of the backplane. However, there was a deeper purpose behind developing the tool. Bertsekas and Gallager of MIT have stated: "The analytical material is used to generate a deeper and more precise understanding of the concepts. Although the analytical material can be used to analyze the performance of various networks, we believe that its more important use is in sharpening one's conceptual and intuitive understanding of the field; that is, analysis should precede design rather than follow it [3]." It is under this light that the value of the software tool can be appreciated.

Chapter 1

Introduction

A new research program, the Photonic Systems and Devices Major Project, has been undertaken at McGill University under the auspices of CITR, the Canadian Institute for Telecommunications Research. The goal of this Major Project is to construct a free-space optical backplane with a terabit/s aggregate bandwidth. Concurrent with its development, it is desired to identify novel, large scale ATM switching architectures made possible by the high connectivity and throughput envisioned for the backplane. Such a backplane would have applications for large scale systems in the fields of information processing, telecommunications and parallel computing [1].

Over more than one year's time, a preliminary conceptual model of the free-space photonic backplane gradually took shape. During its evolution, the need arose for analytic material to model the backplane. Due to the extremely large number of different architectural configurations and operating conditions for the backplane, it became vital to have a software tool which could implement the analytic material.

The goal of the work described herein was to produce this software tool for the purpose of performing an analysis or design space exploration of the backplane. However, there was a deeper purpose behind developing the tool. Bertsekas and Gallager of MIT have stated: "The analytical material is used to generate a deeper and more precise understanding of the concepts. Although the analytical material can be used to analyze the performance of various networks, we believe that its more important use is in sharpening one's conceptual and intuitive understanding of the field; that is, analysis should precede design rather than follow it [3]." It is under this light that the value of the software tool can be appreciated.

1.1 Motivation

The motivation for the project is described below. As clock frequencies and the demand for greater I/O have risen, the limitations to the throughput of electrical backplanes have become apparent. These limitations have provided the impetus for turning to an alternative technology, namely, photonics, to implement backplanes. A review of the some of the past research on free-space optical backplanes is presented following a discussion of the difficulties of electrical technology.

1.1.1 Electrical Backplanes

The industries of computing and telecommunications have seen much growth over the past decade or so. The computing field has moved from single processor machines to massively parallel machines operating on multiple data streams. The telecommunications industry has moved towards integrating voice and data transport functions into one packet switched network in the form of ATM switching. Both of these applications are fueling the demand for high speed backplanes to interconnect large numbers of nodes at very high bandwidths [5].

Current electronic backplanes are capable of very high bit transfer rates while maintaining a reasonable level of interconnect. However, as an ever-increasing number of transistors are integrated onto a single chip, the number of associated pin-outs necessary has not increased proportionately as shown by Rent's rule [2]. With the emergence of very high speed GaAs circuitry, the physical limits of electronic interconnects are being reached [8].

Among the issues that limit the throughput achievable using electrical backplanes are transmission line limitations, crosstalk, clock distribution and skew, power distribution, thermal dissipation [6] and interconnection density. When $v\tau$ (where v is the signal propagation speed and τ is the signal rise time) is comparable to the distances between ICs, transmission lines are necessary for clock and data distribution [9]. At several GHz, losses are on the order of 0.1 dB per cm [9]. Another limiting factor is crosstalk that worsens as the operating frequency increases. The electrical interconnect density is limited by several factors: the number of I/O pads on the chip boundary, minimum spacing required to prevent capacitive and inductive coupling between lines [7] and increasing loss due to decreasing

wire cross-section which increases resistance due to the skin effect at high frequencies.

Among commercial buses, the highest clock rates are in the 100 MHz range with a parallelism of 256 lines giving a 3.2 Gbyte/s throughput at a 5V logic level [11].

1.1.2 Free-Space Optical Backplanes

Due to the aforementioned obstacles optical technology for high speed backplanes has begun to receive considerable attention. The various types of free-space backplanes generally fall into one of three different categories: bulk optical, micro-optical and holographic. The difficulties with these approaches are the packaging of components, alignment accuracy needed and resolution of the optics [7] while the main advantage is the very high interconnect density.

Bulk Optical Implementation

A free-space optical bus, which uses bulk optical components, has been reported in [12]. The interconnects between boards are done with a laser array source which images onto an array of SEED [20] devices. A beam splitter is used to direct the laser array onto the output pad where it is modulated and then reflected back up. The beam splitter then deflects the signal to a mirror that directs the signal down to the input pad of the next board, thus establishing board-to-board interconnect shown in Fig. 1.1.

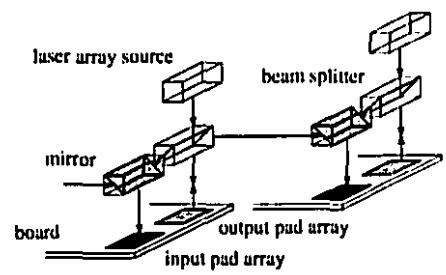


Figure 1.1: Board-to-board interconnect with bulk optics

A five stage free-space optical switching network was reported in [13]. A fully interconnected 32x16 switching fabric was built with an array of SEED devices and macro-optic components. The stages were reported to operate at 50 Mbits/s with 15 active inputs and

outputs.

The main difficulties with these implementations are maintenance of the alignment accuracy and the packaging of such a system for mass production. The connectivity is effectively increased through the use of the third dimension as opposed to two for electronics. A similar scheme was proposed in [14] that used microprism couplers instead of bulk optics to transmit signals between boards with polymer based buses.

Micro-Optical Implementation

Using an array of selfoc microlenses, a free-space optical bus was reported in [15]. The system consists of a backplane into which PCBs can be mounted as shown in Fig. 1.2. The transceivers (photodetectors) on the boards consist of thin layers of amorphous silicon. These are transparent, allowing optical signals to propagate through to all boards.

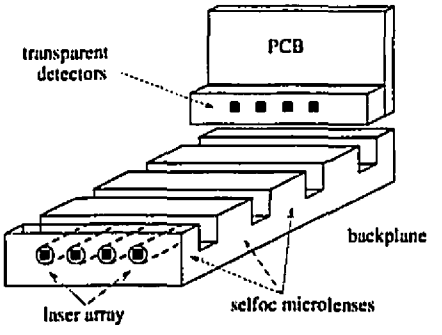


Figure 1.2: Board-to-board interconnect with micro-optics

Using selfoc microlenses of 4 mm diameter, 190x190 channels can be supported in a 2.3 mm square area [16]. Among the advantages are alignment which is simpler than for bulk optical systems due to fewer degrees of freedom, improved thermal stability and a large space-bandwidth product.

Holographic Implementation

Reported in [17] is a holographic method of interconnect between integrated circuits and multichip modules shown in Fig. 1.3. A computer generated holographic plate is suspended 1 mm above the ICs and a mirror 2 cm above that. The GaAs transmitters contain surface

emitting lasers which generate an array of optical signals. These laser arrays pass through sub-holograms which fan-out the signal and reflect off the mirror down through other sub-holograms that act as single lenses to focus the beams onto detectors below. Experiments have shown that a density of over 1100 connections/cm^2 can be achieved. While alignment concerns are less than that of bulk optics, wavelength variations due to temperature and power source fluctuations need to be controlled.

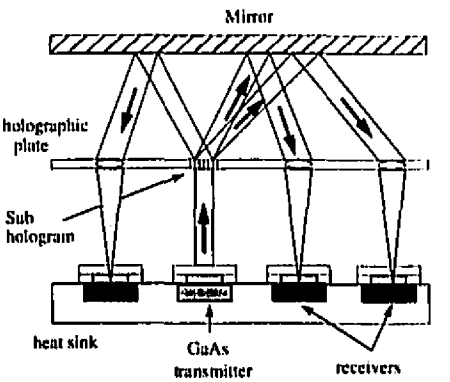


Figure 1.3: Board-to-board interconnect with holographic elements

Even more powerful holographic designs have been proposed in [18], which can be accessed dynamically, where the hologram can support different interconnection patterns within one plate.

1.2 Author's Contribution

The author's contribution to the CITR Photonic Backplane program [1] has been four-fold. The first has been in aiding the development of the analytic model by recognizing mathematical constraints, defining new performance measures, adding refinements and implementing new operational schemes. The second has been in developing a software tool, based on the analytic model, with which a broad design space exploration can be performed. Much effort was devoted to creating a well-structured, hierarchical tool which could be easily revised and expanded in order to accommodate future modifications to the backplane. Much effort was also devoted to optimizing the code in order to provide a tool that is computationally efficient. Having written the software, the third contribution has been in

performing a broad design space exploration with the tool. With the knowledge gained from the exploration, the author has helped to optimize the throughput of the backplane by highlighting its efficient operating point. The last and most important contribution has been the greater conceptual and intuitive understanding gained into the subtleties of the operation of the various hyperplane architectures.

1.3 Overview

In order to describe the software tool, its design goals, hierarchy and its capacity to model the various backplane architectures, the backplane, hereafter called the *hyperplane* [28, 26] is first described. A review of the hyperplane is presented in Chapter 2. The first section contains a description of the structure and the main components. The second section then presents two variations on the basic hyperplane. They are the single-stream circular and dual-stream circular hyperplanes. The third section presents the hyperplane when a number of internally non-blocking networks are embedded. They are the Crossbar switch, the dilated Crossbar switch, the Knockout switch [19], the Fully Connected network, and two novel networks called the Crossout and the dilated Crossout switches [25, 26].

In developing the software, a number of goals had to be set. They described in Chapter 3. The tool should be powerful enough to perform a thorough design space analysis and yet be easy to use. To this end, the high level functionality and user interfaces are described. The software tool is divided into two parts: a numerical analysis section followed by a numerical visualization section, which provides the user with a simple method of viewing the numerical results. Lastly, the hierarchy of the software is presented revealing the ease with which functions can be added or modified. With this logical structure, future variations to the basic hyperplane, operating modes and embedding schemes can be accommodated without difficulty.

The numerical analysis portion of the software is based on a large analytic model, derived in [25, 26] [1] and reviewed in Chapter 4, of the different hyperplanes and networks. The first four sections provide the derivation of the various performance measures related to the unbuffered hyperplane. These measures include, the packet time-slot duration, the

[1] The papers are presently in the journal submission process which typically takes a few years to complete.

probability of packet blocking and network throughput. The last section provides the derivation of the performance measures for the buffered hyperplane (with input queues). These measures include the expected number of packets in the queue and the expected delay. The buffers considered are FIFO input queues of infinite and finite size.

The analytic model of the hyperplane is implemented by a software tool. The author presents a guide to its operation in Chapter 5. The second and third sections describe the numerical analysis and numerical visualization parts of the tool, respectively. The various assumptions made in implementing the model are stated and justified. The last section describes the system requirements and some practical considerations when performing a design space exploration with the tool.

Having developed the software tool, it is used to perform a design space exploration of the photonic backplane. Chapter 6 presents the numerical results of that exploration. A thorough analysis of the linear hyperplane's ability to embed six different networks is presented. A similar analysis is done with three specific cases of the dual stream circular hyperplane. The variations between the different networks simulated by the various hyperplanes under different operating conditions and configurations are shown. As a result of the initial analysis, all causes of any degradation of throughput were found. It is shown that the model provides an excellent understanding of the design space of the hyperplane.

Chapter 7 draws conclusions about the work.

Chapter 2

Overview of the Hyperplane

2.1 Structure

The free-space photonic backplane, presented in [22, 27, 28], is reviewed in this chapter. The backplane consists of four main components: nodes, smart pixel arrays [24], an opto-mechanical support structure and parallel optical channels shown in Fig. 2.1. The individual nodes are printed circuit boards or multi-chip modules which are slotted into the support structure at regular intervals, in an analogous fashion to boards being slotted into an electrical bus.

Communications within a node are electrical, while those between nodes are accomplished through the free-space optical channels in the backplane. These optical channels impinge upon each node's smart pixel array. The array acts as an opto-electronic interface which converts the optical signals into electrical ones for the processing elements of each node and vice versa.

The backplane is envisioned to have between 10 000 and 100 000 parallel optical channels, each carrying a serial bit-stream. These optical bit-channels are to operate between 100 Mbit/s and 1 Gbit/s yielding an overall peak capacity in the terabit/s range. This connection intensive, high capacity interconnection network is called the *hyperplane.*

2.1.1 Parallel Optical Channels

The hyperplane is a linear structure where the N nodes in the backplane are placed in a row. A set of Z free-space optical bit-channels, implemented using laser beams, originates

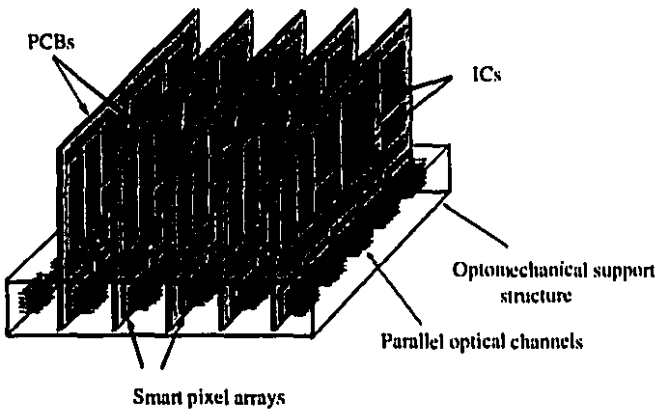


Figure 2.1: Free-space photonic backplane

at node 1 and traverses the backplane to node N. This enables node i to transmit data to node j, where $1 \leq i < j \leq N$. Communications in this direction are known as *upstream*. Similarly, there exists another Z optical bit-channels which originate at node N and traverse the backplane in the reverse direction to node 1. This permits transmissions from node j to node i. Communications in this direction are known as *downstream*.

2.1.2 Nodes

The nodes are composed of two parts, a processing element and a message processor. The processing elements can be general purpose processors such as those found in massively parallel processing machines or more specialized components such as the switching elements found in telecommunication switches.

The optical interface for each node, described below, is implemented with a smart pixel array which can modulate the outgoing optical bit-channels and sense the incoming optical bit-channels. By setting the state of each cell in the smart pixel array, signals or packets of information can be received or transmitted through the backplane.

The message processor is the interface between the processing element and the smart pixel array of a node. It is responsible for implementing the backplane communication protocol by controlling the transmission and reception of data to and from a node. If the backplane is operated with input queues, the message processor has the additional task of managing the queue.

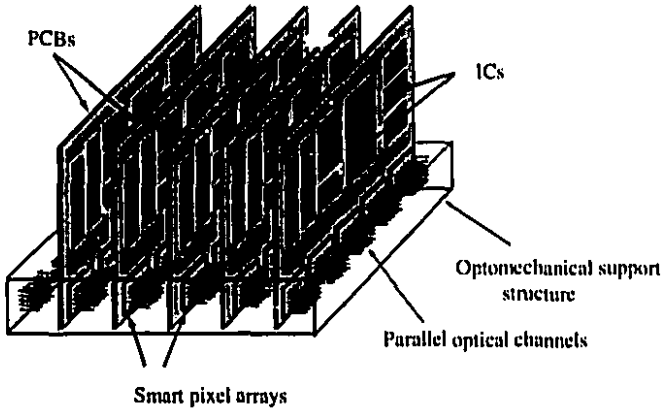


Figure 2.1: Free-space photonic backplane

at node 1 and traverses the backplane to node N. This enables node i to transmit data to node j, where $1 \leq i < j \leq N$. Communications in this direction are known as *upstream*. Similarly, there exists another Z optical bit-channels which originate at node N and traverse the backplane in the reverse direction to node 1. This permits transmissions from node j to node i. Communications in this direction are known as *downstream*.

2.1.2 Nodes

The nodes are composed of two parts, a processing element and a message processor. The processing elements can be general purpose processors such as those found in massively parallel processing machines or more specialized components such as the switching elements found in telecommunication switches.

The optical interface for each node, described below, is implemented with a smart pixel array which can modulate the outgoing optical bit-channels and sense the incoming optical bit-channels. By setting the state of each cell in the smart pixel array, signals or packets of information can be received or transmitted through the backplane.

The message processor is the interface between the processing element and the smart pixel array of a node. It is responsible for implementing the backplane communication protocol by controlling the transmission and reception of data to and from a node. If the backplane is operated with input queues, the message processor has the additional task of managing the queue.

2.1.3 Smart Pixel Arrays

A smart pixel array [24] is composed of an array of smart pixel cells. Each cell is capable of transmitting and/or receiving one bit through the one optical bit-channel that is in its path. The cells are implemented using GaAs based FET-SEED technology [10]. Each cell has a receiver and a transmitter along with some minimal logic. The receiver operates with differential signals and so is composed of two multiple quantum well (MQW) photosensitive diodes [20]. The transmitter is also differential and composed of two MQW diodes which can modulate two incoming laser beams which are then sent out. The cells are grouped into rows of w cells. The rows are referred to as *logical optical communication channels* or simply *logical channels* [24]. Transmission and reception of data are done through these channels, w bits at a time. The logical channels are grouped C at a time to form a *communications slice.* In addition to the C optical channels, a basic slice has two electrical access channels which are w bits wide. One electrical channel is used to receive data from the message processor to be transmitted over a specific logical channel in the backplane. The other electrical channel is used to send incoming data from a logical channel to the message processor. Finally, K slices are assembled to form a complete smart pixel array. A smart pixel array then has a total of $K \cdot C$ logical channels which corresponds to a total of $Z = wKC$ optical bit-channels. A representative example of a basic smart pixel with $w = 8$ bit wide channels, $C = 4$ logical channels per slice and $K = 4$ slices is shown in Fig. 2.2 for a backplane with $Z = 128$ optical bit-channels.

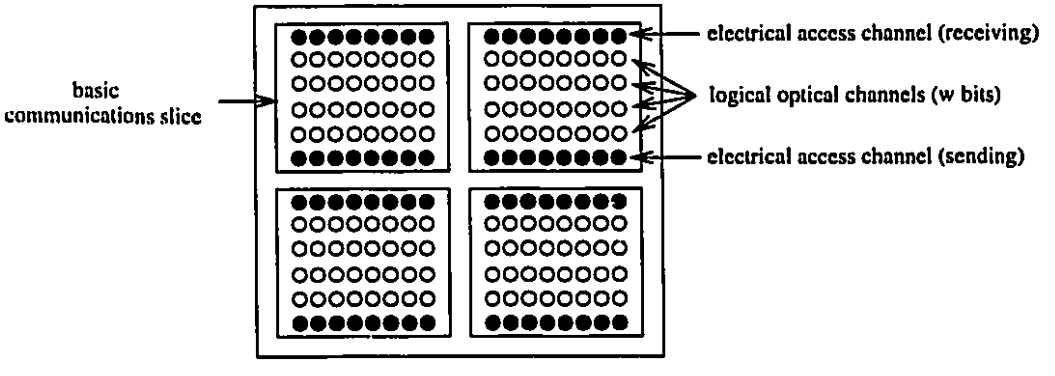


Figure 2.2: A basic smart pixel array

The basic communications slice is able to inject and extract one row at a time. Given a fixed number of optical bit-channels, the number of slices can be varied within a smart pixel array by changing the combination of logical channels per slice and width of the individual channels. Since each slice represents electrical access to the logical channels of that slice, increasing the number of slices per array increases the ratio of electrical I/O bandwidth to optical I/O bandwidth.

Alternately or in addition to increasing the number slices, the capabilities of a slice can be enhanced. The basic slice allows for one transmission at a time. By having s electrical access channels for receiving data from the message processor, multiple rows of data can be sent out through s logical channels. Similarly, the basic slice allows for only one channel out of C incoming logical channels to be received and sent to the message processor. This can be expanded by having b electrical access channels for sending data to the message processor. s is known as the input dilation of a slice and b is known as the output dilation of a slice. Several variations of the basic smart pixel array are shown in Fig. 2.3.

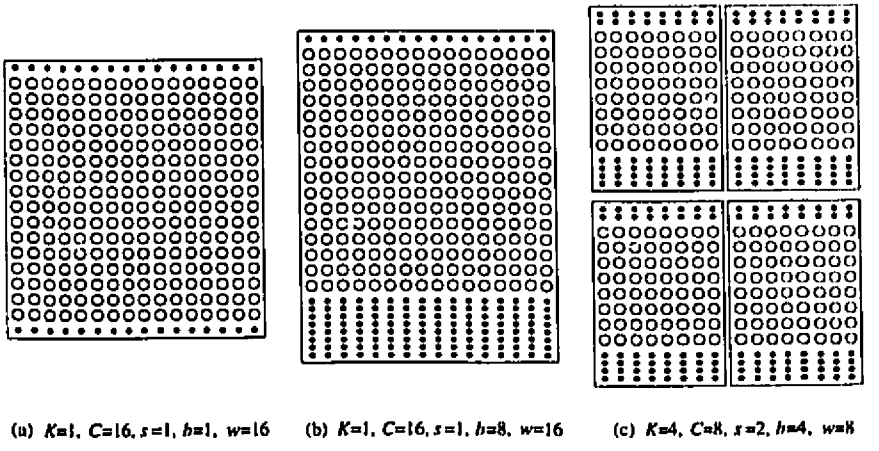


Figure 2.3: Variations of the basic smart pixel array

In the backplane, all nodes have identical smart pixel arrays. The smart pixel array allows a node to access the $K \cdot C$ logical channels in the backplane. The channels are assigned to nodes in such a manner that communication between nodes can take place without collisions. Each node has a sender-reserved channels used for the simultaneous transmission of data. a is known as the input dilation of a node. The first node may

transmit over the first a channels in the first slice, the second node may transmit over the next a channels in the first slice, etc., and the last node may transmit over the last a channels in the last slice. In this manner the a transmitters of the N nodes are assigned to the $K \cdot C$ logical channels. In order to receive data, a node will listen to all $K \cdot C$ channels in the backplane except for its own set of a channels used for transmission.

Various operating configurations of a smart pixel array

The smart pixel array in Fig. 2.3c is worth further consideration. The K slices of an array are always operated in parallel and normally independent of one another. However, this need not always be the case. Each of the four slices has a channel width of eight bits. In practice, the slices can be grouped in pairs to form an array with, effectively, two slices whose channel width is now 16 bits. Instead of pairing the slices, all four slices may be grouped so that the channel width is effectively 32 bits. In general, any number of slices can be grouped to form one effective slice, with a greater channel width.

Instead of, or in addition to, grouping slices, the transmitters within a slice or across slices may be grouped. The slice is set for an $N = 16$ node backplane with $a = 2$ transmitters per node, $K = 4$ slices, $C = 8$ channels per slice and a channel width of $w = 8$ bits. It is possible to change the effective width of a channel, by pairing the two transmitters within a slice to transmit two halves of a packet (a packet typically being several bytes of data). The effective number of transmitters per node then becomes $a = 1$ and the effective number of channels per slice $C = 4$. A wide variety of configurations are possible which can change the effective number of transmitters, slices, channels per slice and channel width. Among the possibilities are $(a = 1, N = 32, C = 8, K = 4, w = 8)$, $(a = 2, N = 8, C = 4, K = 4, w = 8)$, $(a = 4, N = 8, C = 8, K = 1, w = 32)$, $(a = 8, N = 4, C = 8, K = 4, w = 8)$.

2.1.4 Mode of Operation

The hyperplanes described in this thesis are synchronous and operate on a high speed clock. This clock governs the transfer of data between nodes in the backplane. Each optical bit-channel operates at the clock frequency of the backplane. During each clock period, the data on the $K \cdot C$ channels of one smart pixel array is transferred in parallel to a neighbouring

smart pixel array, i.e., from one node to its neighbouring node. While the fundamental unit of time is the period of the backplane's clock, time is grouped into larger units called *packet time-slot durations* [27] made up of several clock periods. During a time-slot, a node is able to transmit an entire packet (normally several bytes) to another node.

2.2 Alternate Structures of the Basic Hyperplane

2.2.1 Single-Stream Circular Hyperplane

The linear hyperplane described so far has two streams for communication and necessarily so. If node i wishes to transmit to node j, where $i > j$, it does so through the upstream channels; to reach node k, where $k < i$, it must use the downstream channels. If there was a link that closed the linear structure, i.e., an optical connection from node N to node 1 then there would be no need for two separate channels. The hardware resources of the two channels could be combined into a unidirectional channel of twice the bandwidth as shown in Fig. 2.4. Closure will require some additional hardware in the form of bulk optics at both ends to steer the light around the loop [28, 26].

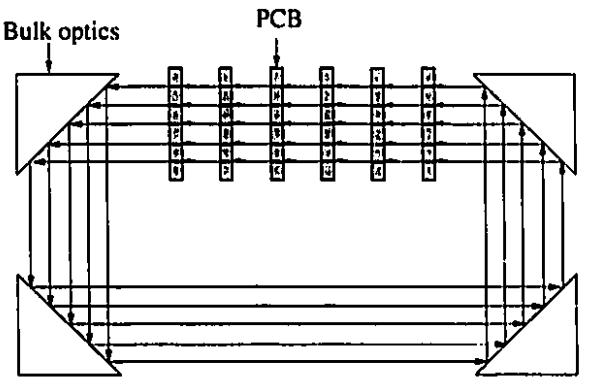


Figure 2.4: Single-stream circular hyperplane

2.2.2 Dual-Stream Circular Hyperplane

Instead of combining the upstream and downstream channels into one unidirectional ring, they can be left as separate streams and configured as two counter-rotating rings [28, 26]. Each ring then has the same number of optical bit-channels as each stream in the linear

version. This configuration has significant advantages over both the linear and single-stream hyperplanes as will be shown in the following section.

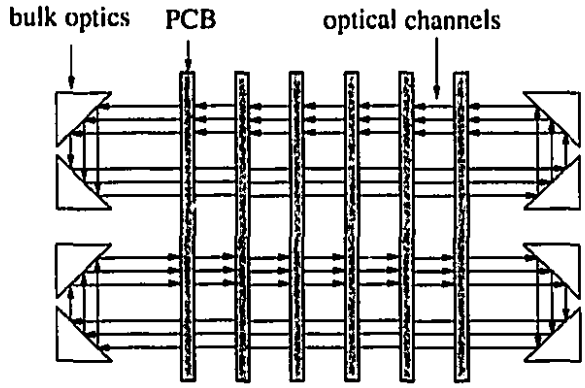


Figure 2.5: Dual-stream circular hyperplane

2.3 Embeddings

One of the strengths of the hyperplane is its connection intensive nature and ability to efficiently embed any other conventional network. The hyperplane can be modeled as a hypergraph with a number of vertices and a large number of edges which qualify the backplane as being connection intensive [25].

A number of common networks are embedded into the hyperplane to demonstrate its performance and flexibility. The networks can be modelled as graphs or hypergraphs [33]. The task of embedding corresponds to mapping the graph or hypergraph of a network onto the hypergraph of the host, being the hyperplane. To illustrate the embeddings, a model or template of the hyperplane is described [21, 25]. It is upon this template, shown in Fig. 2.6 that various networks will be embedded.

The template represents a backplane which can support up to 16 nodes (the nodes correspond to vertices). Each node has a smart pixel array in both the upstream and downstream channels. The arrays, as an example, have $K = 4$ slices and C, an undetermined number of channels per slice. There are $2K$ vertical lines emanating from each node which represent electrical access channels of width w bits.

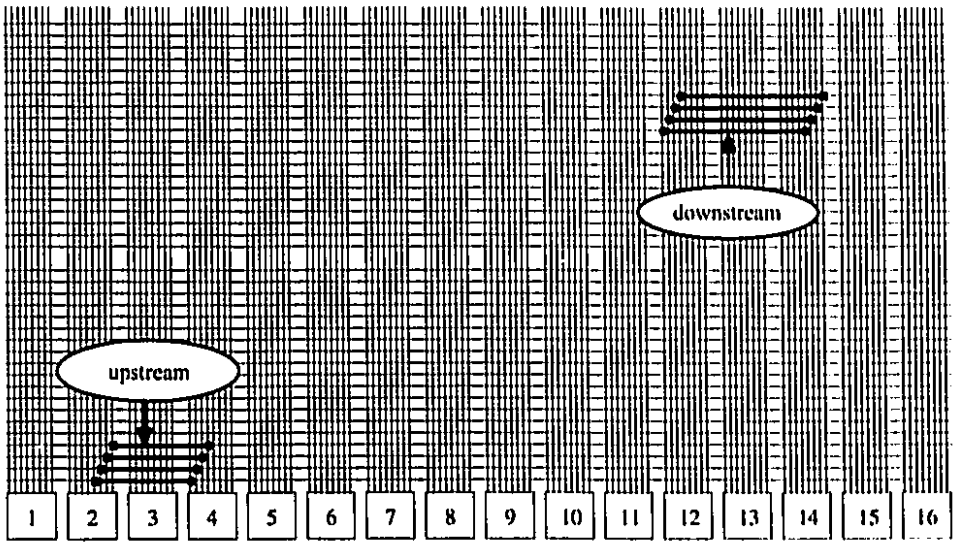


Figure 2.6: Embedding template of the hyperplane

The K vertical lines on the left half of each node represent the electrical channels (one for each of the K slices of the smart pixel array) used to receive packets from other nodes. The packets arrive over the logical channels in the backplane. Each one of the K lines represents $2b$ electrical channels since each slice has b receivers (the factor of two is to accommodate for the two smart pixel arrays necessary for the upstream and downstream communications). The packets received at the slices of a node's smart pixel array are sent to the node's message processor through the electrical channels.

The K vertical lines on the right half of each node represent the electrical channels (one for each of the K slices of the smart pixel array) used to send packets over the logical channels in the backplane to the other nodes. Each one of the K lines represents $2s$ electrical channels since each slice has s transmitters and must accommodate the two smart pixel arrays for the two streams of communication. The packets originate from a node's message processor and are sent through the electrical channels to the slices of the node's smart pixel array.

The horizontal lines represent logical channels that traverse the length of the backplane. The channels are divided into two sections. The upper half is for downstream communications, i.e., from right to left, while the lower half is for upstream communications, i.e., from

left to right. The logical channels are composed of w optical bit-channels, i.e., the same width as the electrical access channels.

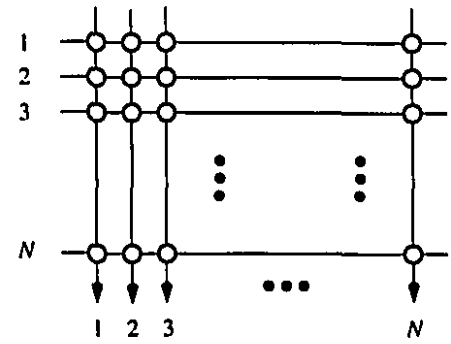
Connections between nodes are represented by bold circles, solid squares and bold horizontal lines. A bold circle represents a logical channel on the slice of a smart pixel array of a node where a packet can be extracted. Similarly, a solid square represents a channel where a packet can be injected into the backplane. A bold horizontal line connects one transmitter (a solid square) to one or more receivers (bold circles) and represents a two-node or multiple node connection.

2.3.1 Linear Hyperplane

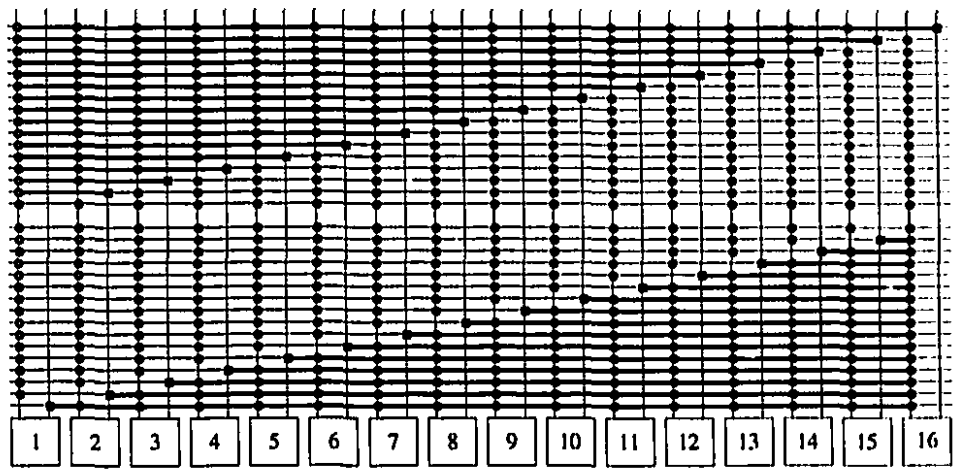
Embeddings in the linear hyperplane are straightforward. There are two streams of Z free-space optical bit-channels each: one for transmissions upstream and the other downstream. Consider one of a edges emanating from node i which spans nodes 1 to N. The portion of the edge which spans nodes $1 \leq j \leq i$ will be embedded into the downstream channels, while the portion of the edge which spans nodes $i \leq j \leq N$ will be embedded into the upstream channels. Thus, all aN nodes will have a part of an edge in both the upstream and downstream channels except for the edges emanating from the outer nodes 1 and N. Since aN edges share Z optical bit-channels, the width of an edge will be Z/aN bits. Consider a transmission of a packet from node 1 to node N. It will take $N-1$ clock cycles for the packet to traverse the intermediate nodes and arrive at node N. This delay is referred to as propagation delay.

Crossbar Switch

The hypergraph model [33] of an $N = 16$ one dimensional Crossbar is shown in Fig. 2.7a. The network has N inputs and N outputs. Each input can send packets over a broadcast bus or hyperedge. Each output may receive from exactly one of the N inputs at one time. The embedding is shown in Fig. 2.7b. There are only two vertical lines from each node which represent the two electrical access channels: one for transmission and the other for reception of packets. Recall that each line actually represents 2 lines to accommodate for the 2 smart pixel arrays necessary for both streams of communication.



(a) Hypergraph model of an N=16 crossbar



(b) Embedding of an N=16 crossbar

Figure 2.7: An $N = 16$ Crossbar switch

Node 1 uses only its upstream transmitters while node N uses only its downstream transmitters.

The smart pixel array of a Crossbar would have $K = 1$ slice and $C = N$ channels per slice. In other words, all the logical channels would reside on one slice. Since the Crossbar has input and output dilations of one, the slice would have $s = 1$ transmitter and $b = 1$ receiver. The logical channels are Z/N bits wide. It is not strictly necessary for the smart pixel array to be configured in exactly this manner. As mentioned earlier, any array that effectively has one slice and a single transmitter and receiver will be suitable.

Knockout Switch

The Knockout switch, proposed by Yeh, Hluchyj and Acampora [19] is similar to the Crossbar but has an N-to-L concentrator on each one of the outputs, where L is the number of receivers per output. The advantage of this scheme over that of the Crossbar is that each output is able to receive up to L packets simultaneously. The hypergraph model is shown in Fig. 2.8.

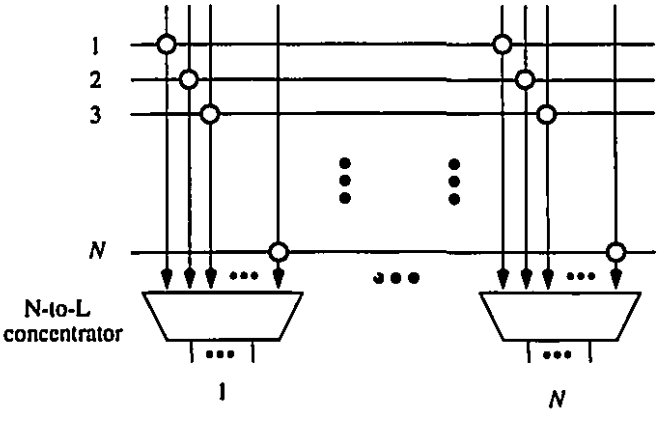
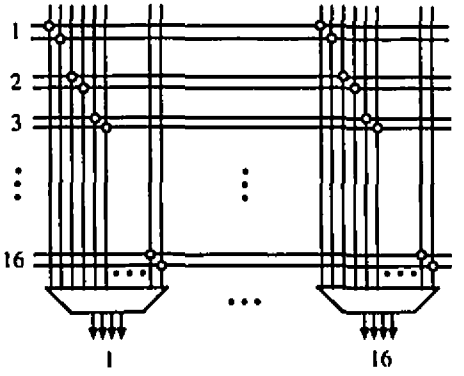


Figure 2.8: Hypergraph model of an $N = 16, L = 8$ Knockout switch

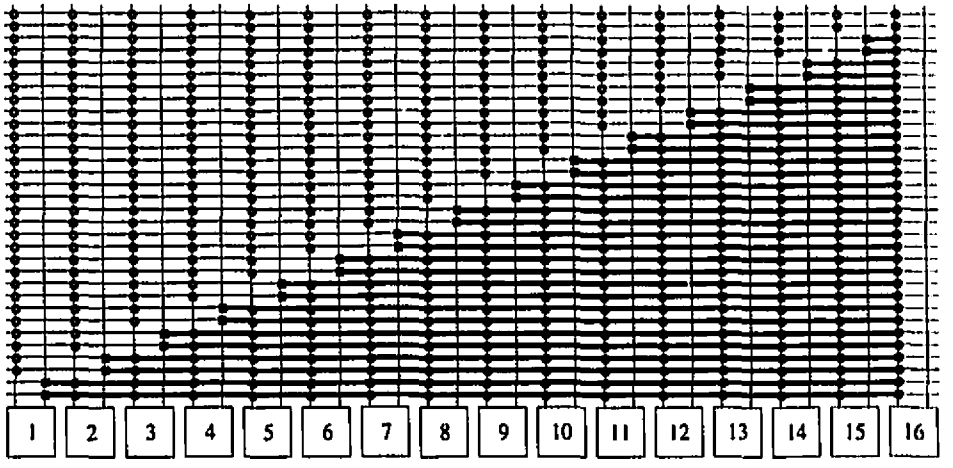
The embedding of the Knockout switch is virtually identical to that of the Crossbar from the point of view of the embedding template. The output dilation is not explicitly shown in the figure. In order to accommodate the L receivers, each vertical line should be considered as being $b = L$ electrical channels operating in parallel instead of a single channel as is the case for the Crossbar and other networks with an output dilation of one. The smart pixel arrays for the Knockout switch would be similar to that of the Crossbar except that the slice must have $b = L$ receivers. The L receivers increase the complexity of the array rather significantly. The logical channels have the same width as in the Crossbar switch.

Dilated Crossbar

The dilated Crossbar is similar to the Crossbar and Knockout switches, except for the multiple transmitters per node. This allows for $a > 1$ packets to be sourced by each input.



(a) Hypergraph model of an N=16, a=2, b=4 dilated crossbar



(b) Embedding of an N=16, a=2, b=4 dilated crossbar

Figure 2.9: An $N = 16, a = 2, b = 4$ dilated Crossbar switch

Each output has an aN-to-b concentrator which allows for the reception of up to b packet simultaneously. The hypergraph model of a dilated Crossbar with $a = 2$ transmitters and $b = 4$ receivers per node is shown in Fig. 2.9a. The upstream channels of an embedding are shown in Fig. 2.9b; the downstream channels would be similar, just as the two streams of the Crossbar are symmetric. As with the Knockout switch, the output dilation cannot be explicitly seen in the embedding. The vertical lines should be considered as b channels emanating from the smart pixel array to the node's message processor.

The smart pixel array for the dilated Crossbar is similar to that of the Crossbar and Knockout switches but has intermediate complexity. The array would effectively have one

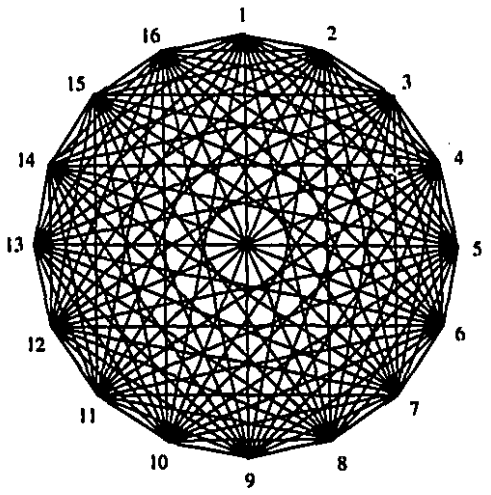
slice containing $C = aN$ channels per slice, b receivers and $s = a$ transmitters. The number of edges to be embedded is aN as opposed to N for the Crossbar and Knockout switches. Consequently, given a fixed number of optical bit-channels, the width of the logical channels will be reduced by a factor of a, i.e., $w = Z/aN$ bits, to accommodate for the input dilation.

Fully Connected Network

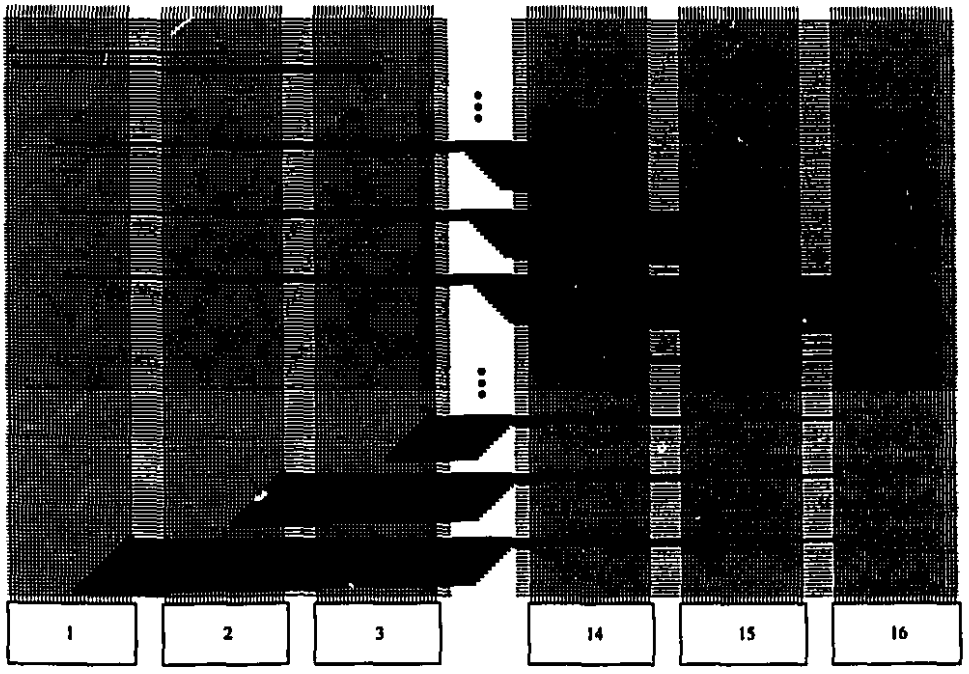
A graph model of an $N = 16$ Fully Connected network is shown in Fig. 2.10a. Unlike the other networks presented in this chapter, the Fully Connected network is a graph and not a hypergraph. The edges are direct links between exactly two vertices (or nodes) and not broadcast links such as those in the hypergraph networks. Every node has an edge to every other node in the network. The embedding is shown in Fig. 2.10b. Since every node is directly connected to every other node, there must be $a = N$ transmitters per node. This is implemented using a smart pixel array with $K = N$ slices, $C = 1$ channel per slice and $s = 1$ transmitter per slice. Node 1 uses 15 upstream transmitters and none on the downstream array while the opposite is true for node N. The intermediate nodes, $1 < i < N$, use $N - i - 1$ upstream transmitters and i downstream transmitters. This means that $N + 1$ transmitters per node remain unused, i.e., just over half which represents a significant inefficiency in hardware utility. The same could be accomplished by having an array of $K = 1$ slice, $C = N$ channels per slice and $s = N$ transmitters for the slice. As N grows large, the number of transmitters per node will be limited to some number less than N due to hardware costs. The transmitters can be mapped to several outgoing channels to maintain full connectivity. However, each node will only be able to source a limited number of packets less than N. Since there are $N(N-1)/2$ edges embedded in Z optical bit-channels, the width of an edge is $2Z/(N(N-1))$ bits.

Crossout Switch

The Crossout switch is a novel network proposed in [25]. Its hypergraph model is shown in Fig. 2.11a. It is similar to the Crossbar and Knockout switches but unlike the latter which has a single N-to-L concentrator per output, the Crossout switch has K concentrators per output each of size N/K-to-b and has intermediate complexity. Thus each concentrator



(a) Graph model of an N=16 Fully Connected network



(b) Embedding of an N=16 Fully Connected network

Figure 2.10: An $N = 16$ Fully Connected network

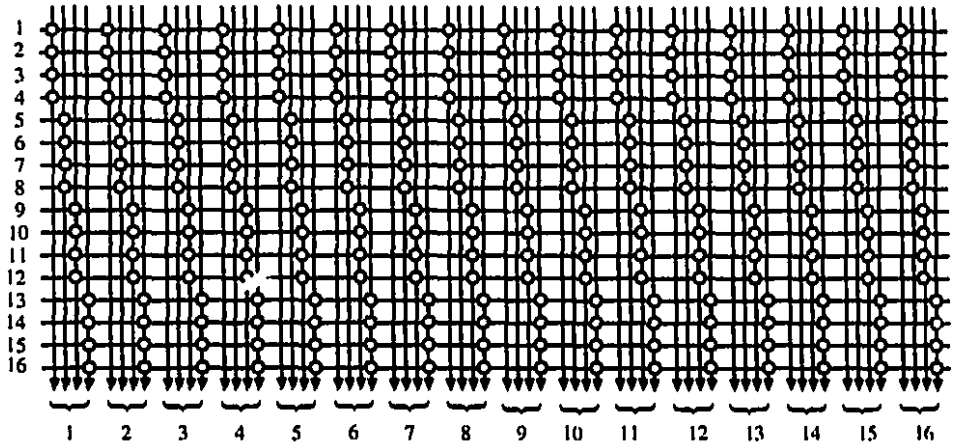
operates on a subset of the N incoming logical channels. The embedding is shown in Fig. 2.11b. The smart pixel array used for each node has $K = 4$ slices, $C = 4$ channels per slice, $s = 1$ transmitter and $b = 1$ receiver per slice. Since the Crossout switch has N edges, the width of the logical channels is Z/N bits.

Dilated Crossout Switch

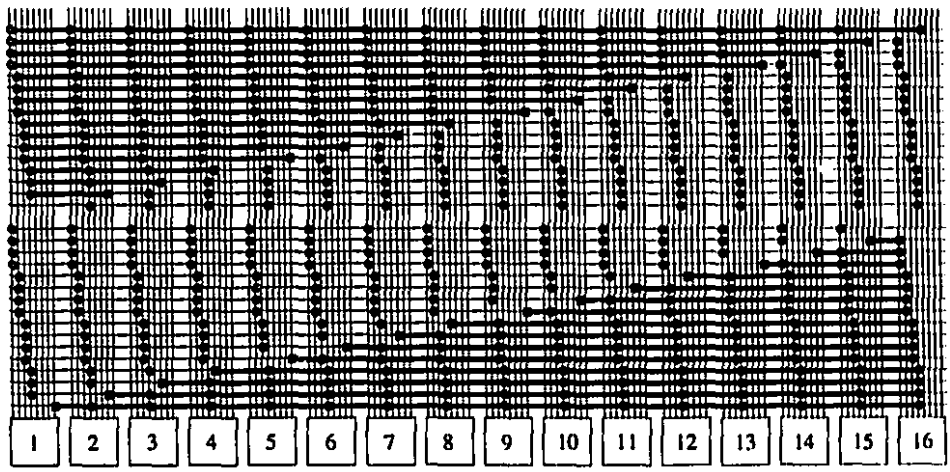
The dilated Crossout switch is similar to the Crossout switch, except for the multiple transmitters per node. This allows for $a > 1$ packets to be sourced from each node. The graph model of a dilated Crossout switch with $a = 2$ transmitters and $b = 4$ receivers per node is shown in Fig. 2.12a. The embedding is shown in Fig. 2.12b. The smart pixel array used for each node has $K = 4$ slices, $C = 4$ channels per slice, $s = 2$ transmitters and $b = 8$ receivers per slice. Given the same number of optical bit-channels, the width of the logical channels will be reduced by a factor of a over that of the Crossout switch to accommodate the dilation. As with the Knockout switch, the output dilation cannot be explicitly seen in the embedding. The four vertical lines on the left half of each node should be considered as b channels emanating from the smart pixel array to the node's message processor.

2.3.2 Dual-Stream Circular Hyperplane

Embeddings in the dual-stream circular hyperplane can be accomplished in three distinct manners [26]. As with the linear hyperplane, there are two streams of Z free-space optical bit-channels each; one for transmissions upstream and the other downstream. However, each stream is actually a ring due to closure of the ends. Nodes are then able to send packets over the N-to-1 link which did not exist previously. This capability, which is not present in the linear hyperplane and may not seem significant, leads to the different embedding schemes. Three schemes are presented below for the Crossout switch, of Fig. 2.11, to illustrate the impact of the dual stream hyperplane. Although not shown explicitly, these embedding schemes apply equally to the other networks presented above.

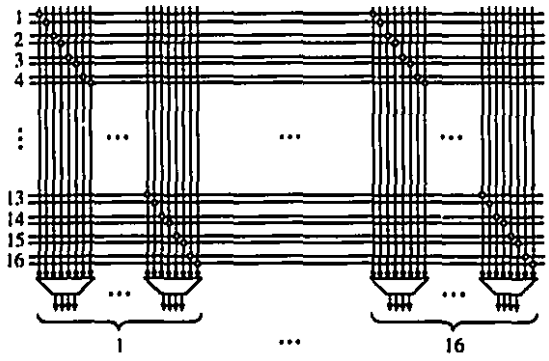


(a) Hypergraph model of an N=16, C=4, K=4 Crossout

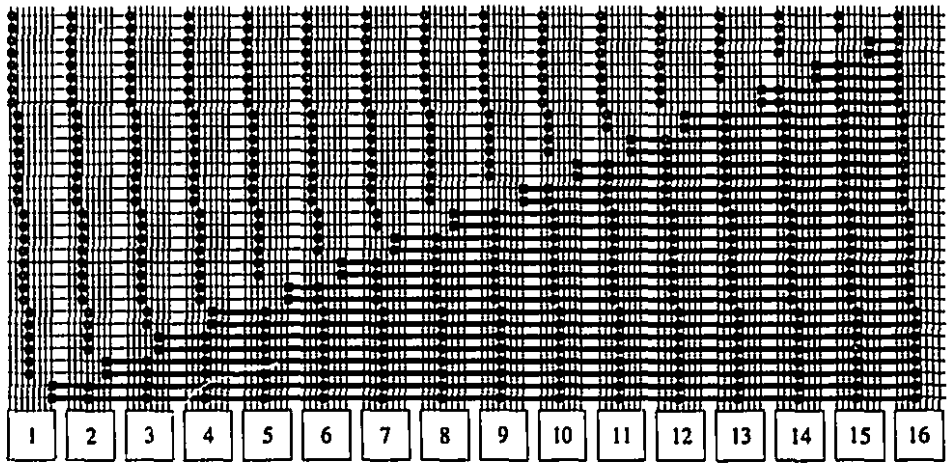


(b) Embedding of an N=16, C=4, K=4 Crossout

Figure 2.11: An $N = 16, C = 4, K = 4$ Crossout switch



(a) Hypergraph model of an N=16, C=8, K=4, a=2, b=4 dilated crossout



(b) Partial embedding of an N=16, C=8, K=4, a=2, b=4 dilated crossout

Figure 2.12: An $N = 16, C = 8, K = 4, a = 2, b = 4$ dilated Crossout switch

Maximized Edge Bandwidth Embedding

In the first scheme, the embedding can be done such that the channel width or edge bandwidth is maximized. By embedding half of the aN edges of a network in one ring and the other half in the other ring, the aN nodes will share a total of $2Z$ bit-channels. The channel width is then $2Z/aN$ bits. In the worst case, the propagation delay will remain to be $N-1$ clock cycles since transmissions from node i to node $i-1$, where $i < N/2$, in the upstream channels will require traversing $N-1$ other nodes. The same is true in the downstream channels for transmissions from node $i-1$ to node i where $i > N/2$.

An embedding of an $N = 16, C = 4, K = 4, a = 1, b = 1$ Crossout switch where the

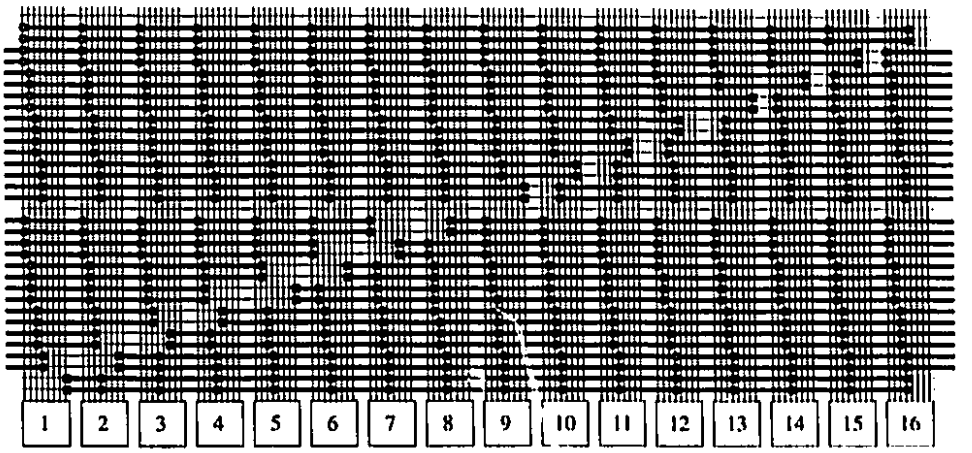


Figure 2.13: Maximized edge bandwidth embedding of a Crossout switch

edge bandwidth has been maximized is shown in Fig. 2.13. Despite the two transmitters per node in the diagram, they are effectively one transmitter. They should be thought of as two halves of one transmitter with an effective width which is twice that of transmitters of the Crossout switch in the linear hyperplane shown in Fig. 2.11. In order to accommodate the two coupled transmitters, the number of receivers per slice must be at least two. These two receivers are effectively one receiver of twice the channel width as a single receiver.

It should be noted that this embedding corresponds to the single-stream circular hyperplane with $2Z$ optical bit-channels. Thus, the single-stream circular hyperplane, offers no advantages over the dual-stream architecture.

Minimized Propagation Delay Embedding

In the second scheme, the embedding can be done such that the propagation delay is minimized. An embedding of an $N = 16, C = 4, K = 4, a = 1, b = 1$ Crossout switch under this scheme is shown in Fig. 2.14. By embedding all the aN edges in both rings, transmissions never have to traverse more than $N/2$ intermediate nodes. Consider a transmission from some node i to some other node j, where $1 \leq i < j \leq N$. If the distance in one ring is $d1 > N/2$, then the distance in the other ring will be $N - d1 < N/2$. Since an edge from a given node always exists in both rings, a transmission some node i can be sent out using that ring in which the distance is shorter. In the worst case, the propagation delay will be

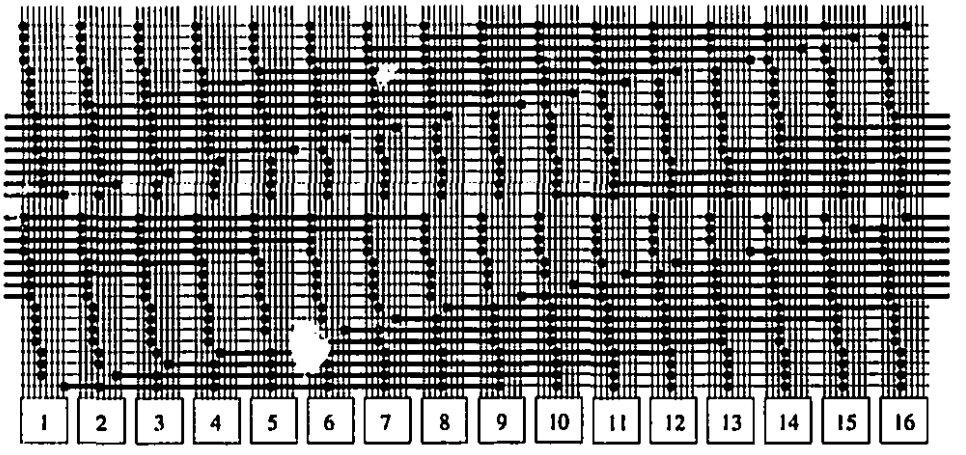


Figure 2.14: Minimized propagation delay embedding of a Crossout switch

$N/2$ clock cycles. Since all aN edges are embedded into each of the Z optical bit-channels of both rings, the width of the logical channels is Z/aN bits per edge.

Maximized Edge Bandwidth and Minimized Propagation Delay

In the third scheme, the embeddings can be done such that edge bandwidth can be maximized and the propagation delay minimized simultaneously. An embedding of an $N = 16, C = 4, K = 4, a = 1, b = 1$ Crossout switch where the bandwidth has been maximized and the propagation delay minimized is shown in Fig. 2.15. Consider the second scheme, described above, which minimizes the propagation delay. Each embedded edge spans $N/2$ nodes. Half of each logical channel remains unused since there is no edge or portion of an edge which occupies it. These edges can be packed twice as densely so that two edges occupy the same set of logical channels without overlapping, e.g., from node 1 to node $N/2$ and from node $N/2+1$ to node N. The aN edges would now occupy only half of the logical channels in each ring. This now means that the width of each edge can be doubled by allocating two logical channels instead of one per edge. In this manner, the edge bandwidth is maximized, i.e., $2Z/aN$, while maintaining the minimized propagation delay of $N/2$ clock cycles.

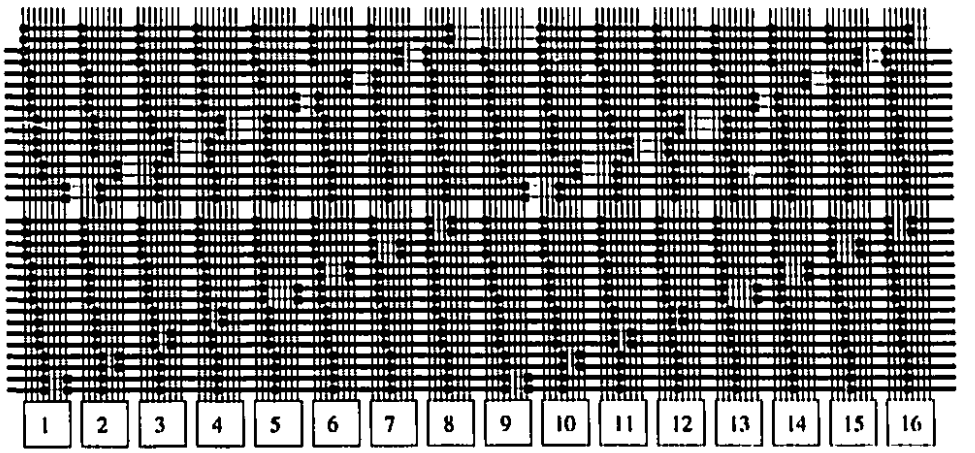


Figure 2.15: Optimized bandwidth and propagation delay embedding of a Crossout switch

2.4 Summary

The hyperplane was reviewed in this chapter. Its structure and main components were described in the first section. Two variations of the basic linear hyperplane, the single-stream and dual-stream circular hyperplanes, were then described in the second section. In order to illustrate the connection intensive nature of the hyperplane, a number of common networks were embedded in the linear hyperplane. Finally, three special embedding schemes which apply to the dual-stream circular hyperplane were described.

Chapter 3

Design Goals of the Software Tool

In designing the software tool, a number of goals were kept in mind. As stated by Bertsekas and Gallager, analysis should precede design rather than follow it [3]. The motivation behind this is that a thorough understanding of network or system operation should be gained prior to any attempt being made to design and optimize a network based on educated guesses or past experimentation with similar systems [4]. This understanding removes some of the uncertainty which is inherent in the design process. Thus, above all else, the fundamental goal in designing the software is to provide a tool which will allow the network designer or system level architect to develop an intuitive understanding for the effects that various parameters have on design and operation. Given this understanding, a network can be developed where the available hardware resources are used efficiently and the ideal operating point has been selected which meets all applicable constraints related to technology and specifications related to standards. It can then be stated with reasonable confidence that the highest level of performance possible has been achieved.

Having established the primary goal, a number of secondary goals which are necessary to build the software tool are identified. Due to the complexity of the hyperplane there are many differing configurations related to architecture, various possibilities for network embedding schemes and an infinite range of operating points and conditions under which it may function. To fulfill its role as a tool capable of being used for broad design space explorations, the software must have a flexible mathematical model to analyze the numerous incarnations of the hyperplane. In developing the model, the suitable level of abstraction needs to be determined. For the system level architect minute or very intricate details are

neither needed nor desirable as they would detract from the main purpose of modeling at the system level. Along the same lines, there are a large number of performance measures which characterize all aspects of the operation of the hyperplane. Only a subset of those which are of relevance are actually calculated.

In writing the software code to implement the mathematical model, steps must be taken to ensure that the computations are relatively unintensive yet still be sufficiently accurate so that incorrect conclusions are not drawn. At the top level, all of the individual routines should be assembled in such a way that the software has a sound logical internal structure. This would allow for future innovations, be they related to the architecture, device technology or the operating conditions, to the hyperplane to be accommodated. Finally, the user of the tool should not be unnecessarily burdened with the underlying functions, file handling etc., of the software. A suitable front end which hides these aspects is needed. The following sections in this chapter discuss in greater detail these goals.

3.1 Hardware Configurations

Firstly, the software must model two different types of hyperplanes: the linear and dual-stream circular. The single-stream circular is not modeled since it offers no advantages over the dual-stream version. The dual-stream circular hyperplane under the maximized edge bandwidth embedding scheme is equivalent to the single-stream circular hyperplane.

The linear hyperplanes can be operated in two different modes. Recall that each node transmits over a channels which are reserved for it. For each node, the assignment of the subset of the a transmission channels to the available aN channels is arbitrary. Two particular *channel assignment* schemes, one denoted *sequential* and the other *interleaved*, presented in the following chapter, are considered. The software should model the impact of these two schemes.

The dual-stream circular hyperplane has some advantages over the linear version. As outlined in the previous chapter, three different embedding schemes are possible for the networks being considered. By varying the manner in which edges of a network are embedded into the two counter-rotating rings, the bandwidth can be maximized, the propagation delay minimized or both achieved simultaneously. The software should allow for each different

case to be modeled.

Second, the software must model two parameters, related to the hardware, which define the capacity and scale of the hyperplane. The first parameter is the number of free-space optical bit channels denoted by Z. For either hyperplane, the peak capacity is directly proportional to the number of optical bit channels. The capacity is limited by the optical hardware used to generate the parallel beams of light, where each beam carries a single bit. As optical technology improves, the density and number of optical bit channels will increase. The second hardware parameter is N, the network size given by the number of nodes. Scalability of the architecture is an important concern as future systems may require ever increasing number of nodes to be supported by the backplane. The software should be able to model the effects of incorporating varying numbers of nodes into the backplane.

3.2 Operating Conditions

Once the hardware configuration is decided upon, there are three parameters which affect the operation of the hyperplane which the software should allow to be altered and modeled.

The clock frequency, B, of the backplane determines the rate at which bits are transferred from one smart pixel array to an immediately neighbouring array, i.e., between adjacent nodes. The operating frequency of the clock depends upon the maximum rate at which the optical receivers in the smart pixel array are able to switch. Presently, GaAs/AlGaAs multiple quantum well diodes are being used in the smart pixel arrays as receivers. As the fabrication process matures or alternate elements are used, the maximum clock rate will change. The software should model operation of the backplane at any clock frequency.

Transfers between nodes are done using a fixed packet size. For ATM standards a cell is defined as 53 bytes long. The packet size, P, directly affects the time duration of a time-slot, which in turn affects the throughput of a network. The software should model transmission of packets of any arbitrary size in bits.

Backplanes, whether part of a telecommunications switch or the interconnection network of a multi-processor machine, will face varying loads at different times. Ideally, the hyperplane should be able to carry traffic regardless of the offered load, denoted α, without significant loss of packets. The software should be able to model the hyperplane under any

traffic load.

3.3 Performance Measures

Any software tool which is to be used for a design space exploration for the development of the hyperplane must compute and present all the relevant performance measures. The optical backplane being an entirely new endeavour, there are a large number of characteristics to be explored. The relevant aspect of the hyperplane must be examined in order to achieve a thorough understanding of its operation. These measures should be computed as a function of the number of nodes in the backplane, the number of optical bit-channels, the operating clock frequency, packet size and the traffic load where appropriate. The hardware configurations and operating conditions can then be set so that the best performance can be obtained for various applications. The software models the hyperplane under two conditions: unbuffered and buffered.

3.3.1 Unbuffered Analysis

In the unbuffered analysis, several important measures are identified, along with their related ones, which the software should compute. These are the packet time-slot duration, the probability of packet blocking, the loss rates and throughput.

The packet time-slot duration, described in Chapter 4, is a crucial parameter which affects the throughput of networks embedded in hyperplane. It is dependent upon the type of hyperplane and the particular embedding scheme being used.

All of the networks examined here are internally nonblocking. In the presence of permutation traffic no blocking occurs at any of the output ports since there is never more than one packet destined for any one output. However, given random traffic, the possibility that multiple messages are destined for the same output port or node exists. The probabilities of blocking and acceptance in the circular hyperplanes are dependent upon the embedding scheme and are different from that of the linear hyperplane.

The final measure is the capacity of the networks embedded in the hyperplane to carry traffic. Among the aspects to be examined are the aggregate bandwidth, the node bandwidth, the edge bandwidth, the unused capacity of the backplane and the loss rate due to

blocking of packets at the output ports.

3.3.2 Buffered Analysis

Networks, such as the hyperplane, may be operated with some type of buffering, in practice. The software should be developed with the capability to model the hyperplane's performance when nodes are buffered with FIFO input queues.

The software models two types of queues, a hypothetical infinite size queue and a realizable one with queues of finite size, Q_s. The parameters of interest are the throughput of the queues, the expected number of packets in the queue, the expected delay of a packet through the queue. For the finite size queue, losses can occur when the queue becomes full, hence, the probability of packet loss and the packet loss rate are of examined.

3.4 Modeling Accuracy and Computation Complexity

For the software tool to be practical, a balance must be found between modeling accuracy and computation complexity. A very low level of abstraction will result in a highly accurate model of the hyperplane but will incur the penalty of excessively long computation time. Since this tool is intended for the system level designer, a number of simplifications can be made which will not unduly compromise the correctness of the analytic model but allow for significant savings in computation time. The tradeoffs used to achieve this goal for the unbuffered and buffered analysis of the hyperplane are presented below.

3.4.1 Unbuffered Analysis

In considering the unbuffered analysis, the bulk of the computational complexity will lie within the functions needed to calculate the probability of packet blocking and acceptance. These probabilities are modeled exactly using binomials and involve large summations, as will be shown in the following chapter. While this binomial model is exact, the computation time can become exceedingly long. This difficulty can be addressed in a couple of different ways.

One approach is to truncate the computation of the probabilities once a certain error tolerance has been reached. In this way, the computation time can be drastically shortened.

At the same time, it can be guaranteed that the result is accurate within a certain, known, percentage. The other approach is to approximate the binomial model with a Poisson model. It provides a less accurate analysis but is computationally much less intensive. The summations have finite bounds in practice and avoid using combinations.

It has been decided that the software should allow the user to select between three different probability or computational models with an increasing speed versus decreasing accuracy tradeoff, the first being an exact binomial model, the second an approximate binomial model which uses truncation and the third being a Poisson model.

3.4.2 Buffered Analysis

In the software tool, the hyperplane is operated in a synchronous fashion; packet transmissions are initiated only at the beginning of a packet time-slot duration. The transmissions are governed by the clock which controls transfers of packets between nodes in the backplane. When analyzing input queues, the appropriate model would then be a discrete-time Markov chain. As is well known, this type of analysis requires solving systems of linear equations for the state of a queue [25].

A much more tractable solution is to analyze the input queues through the use of continuous-time Markov chains. The solutions to the state of the queue are well known closed form equations. The continuous-time model assumes that the inter-arrival times of packets to a queue are random and occur throughout a continuous time period. Despite this misrepresentation, the model provides a very good approximation to the behaviour of the queue over time periods which are much greater than one packet time-slot duration. With this approach the goal of keeping the software routines for the input queuing analysis relatively simple can be met.

3.5 Software Interface

In order that the software tool be simple to use yet powerful enough to carry out a thorough and wide design space exploration, the interface should meet certain criteria. It should require little or no knowledge from the user of the internal software functions or structure. This goal is achieved with an intuitive, fully automated menu driven interface which hides

the underlying operation. The second criterion is that the interface provides full control over the hardware configuration, operating conditions and the computation model.

An obvious division of the software is into two parts according to the intended function. The first part is an interface which allows the user to set up an analysis of the hyperplane under a particular hardware configuration and a set of operating conditions. The second part is an interface to control the visualization of the numerical results.

3.5.1 Numerical Analysis Interface

As stated previously, the design space for the development of the hyperplane is large. Of the large number of the parameters which characterize the hyperplane, several of them are interdependent and cannot be set arbitrarily. Here, the goal is to provide the user with an interface that is intelligent and presents all the parameters in an organized manner. A sample of one of the windows used for the numerical analysis interface is shown in Fig. 3.1. This particular window allows the user to set the packet size, queue size, clock frequency, number of optical bit-channels, etc. With similar windows that complete the interface, the user is provided with full control over the architecture type, network embedding scheme and the computational model.

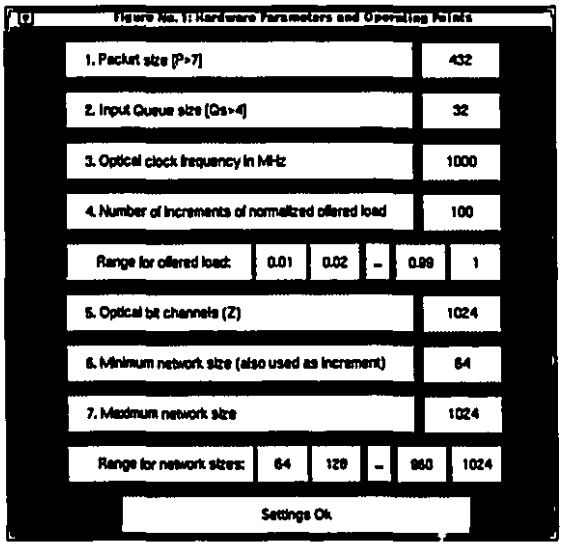


Figure 3.1: A window of the numerical analysis interface

3.5.2 Numerical Visualization Interface

The numerical analysis section encompasses over 24 different performance measures, most of which are computed over a three dimensional design space. The goal in developing the numerical visualization interface is to facilitate handling the large amount of data generated. The interface provides a set of windows which allows the user to specify a particular range over which graphing should be done and then select a particular measure to be graphed for each of the networks. A sample window of the numerical visualization section of the software tool is shown in Fig. 3.2. The window is used for plotting the performance measures calculated in the numerical analysis section. The window includes a set of interactive controls among which include options for rescaling and resizing the axes. Details on the functionality of the visualization part of the tool are provided in Chapter 5.

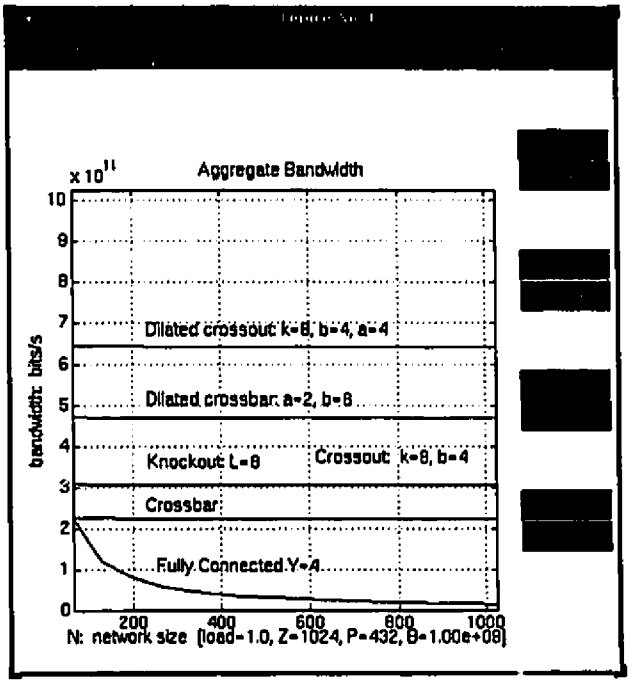


Figure 3.2: A window of the numerical visualization interface

3.6 Hierarchical Design

The software can very rapidly lose its usefulness as a tool for design space exploration if it cannot model future innovations to the hardware and changes to the operating modes and conditions.

In order to delay the eventual obsolescence of the tool, the software should be written in a highly structured and hence hierarchical manner. The goal is that changes to the technology can be easily reflected in the software by modifying the particular function or functions which model the change. The software must be carefully structured so that changes can be made to specific routines which have a global effect. In this manner an update need not be made over a large number of routines and can be kept relatively isolated.

3.6.1 Numerical Analysis Software Structure

The structure of the numerical analysis section of the software tool is shown in Fig. 3.3. The

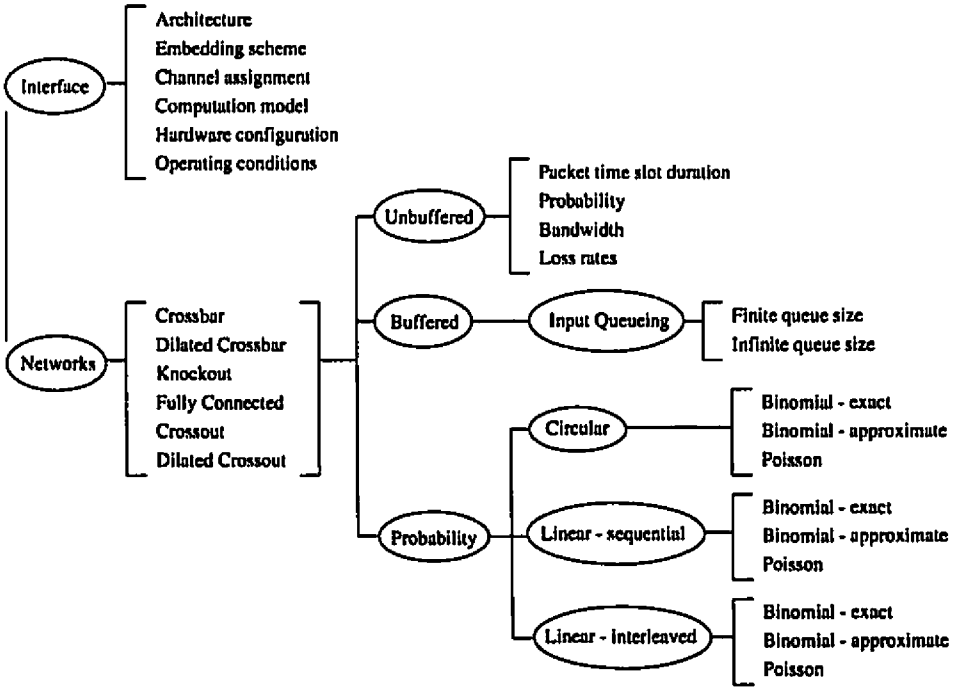


Figure 3.3: Structure of numerical analysis section of the software tool

first half under **Interface** gives the user complete control over the parameters of the design

space exploration to be performed. All of these parameters are organized into different sections. The second part under **Networks** contains the routines used to calculate the various performance measures for each one of the embedded networks. The analysis is divided into three sections: unbuffered and buffered operation, and probability routines. The unbuffered section contains a set of functions which calculate various measures such as bandwidth. The buffered section contains the routines for analyzing performance under input queuing. The probability section is subdivided into three parts to accommodate three different models: circular, linear-sequential and linear-interleaved. Within each one of these, three different probability computation models are stored.

This hierarchy facilitates future revisions and additions to the software tool. New networks, output queuing, different blocking probability models, other performance measures, etc., may be added with ease. Even different hardware configurations, embedding schemes, operating modes and points can be analyzed by adding or modifying the necessary functions.

3.6.2 Numerical Visualization Software Structure

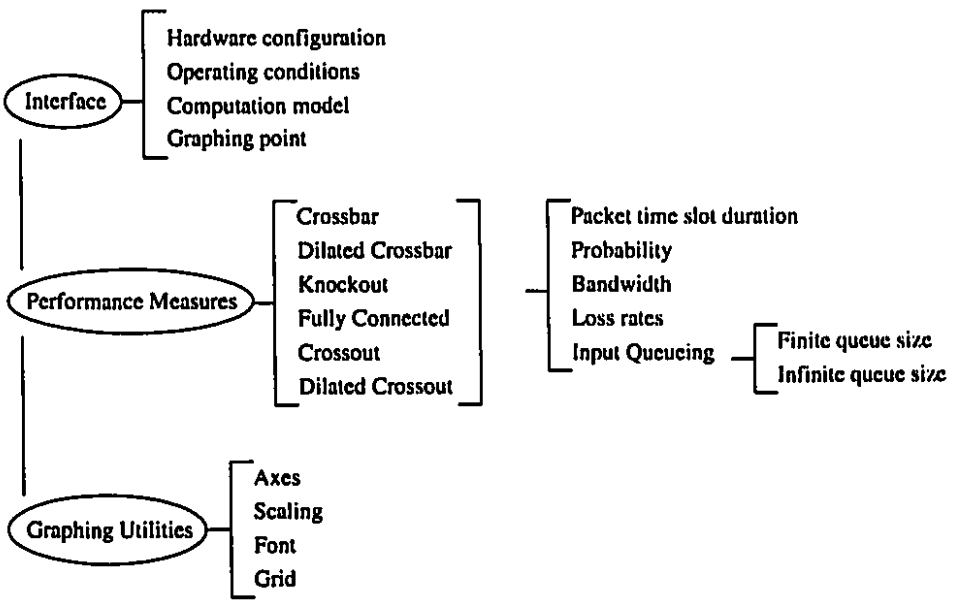


Figure 3.4: Structure of the numerical visualization section of software tool

The structure of the numerical visualization section of the software tool is shown in

Fig. 3.4. The first part under **Interface** organizes the routines for displaying the parameters of the last analysis in a logical manner. The second part under **Performance Measures** organizes the performance measures that were calculated under logical sections. The last part under **Graphing Utilities** stores the graphing utilities under appropriate sections.

As with the numerical analysis structure, the goal of having implemented such a structure is to allow future revisions and additions to the tool. The addition of new networks, performance measures, three dimensional plots, graphing utilities, etc., can be done by simply adding routines under the appropriate sections.

3.7 Summary

The design goals and considerations of the software tool were presented in this chapter. In order for the software to be useful as a tool for performing design space explorations, the tool must be capable of modeling the numerous architectural configurations and operating points of the backplane. The tool must then present the results in an informative manner so that an insight can be gained into the operation of the backplane.

Chapter 4

Analytic Model of the Hyperplane

In this chapter, the mathematical model and various performance measures for the hyperplane, which was presented in [25, 26], is reviewed. The author then identifies some modifications to those measures that allow for a somewhat different perspective on the operational characteristics of the hyperplane. The author also presents a novel embedding scheme and accompanying analysis for the Crossout and dilated Crossout switches.

4.1 Theoretical Performance Analysis

A generalized model for the dilated Crossout is derived in first five sections. Specific instances of this model are derived in order to provide models for the Crossbar, dilated Crossbar, Knockout, Crossout switches and the Fully Connected network. To develop the generalized model, the following parameters are defined.

- Z = the number of free space optical single bit-channels in each direction, upstream and downstream
- P = the size of a packet in bits
- B = the bit rate of the backplane
- N = the number of nodes in a network
- K = the number of communications slices on a smart pixel array
- C = the number of logical channels on a slice

- α = the probability a node has a packet to send in a time-slot
- a = the input dilation of a network
- s = the number of transmitters per slice
- b = the number of packets which a *slice* can simultaneously extract
- e = the number of edges in a network

4.2 Packet Time-Slot Duration

4.2.1 Embeddings in the Linear Hyperplane

The packet time-slot duration [27] is the time during which a node may transfer a packet to another node. The time-slot is composed of two parts: the transmission delay and the propagation delay. The transmission delay, T, given in Eqn. 4.1, is the time required for a node to inject a P bit packet into the backplane over an edge of width Z/e bits. Once injected into the backplane, the packet must travel, through intermediate nodes, over the edge to its destination node. In order to accommodate the worst case, i.e., transmission between the two furthest separated nodes, the propagation delay is set to $(N-1)/B$ seconds. The packet time slot duration, S, is given in Eqn. 4.2.

$$T = \frac{\left\lceil \frac{Pe}{Z} \right\rceil}{B} \text{ seconds} \tag{4.1}$$

$$S = \frac{\left\lceil \frac{Pe}{Z} \right\rceil + (N-1)}{B} \text{ seconds} \tag{4.2}$$

The propagation delay portion of the packet time-slot duration represents an inefficiency in throughput. During that time, the sending node is idle and no longer transmitting. Consequently, the channel is partially unused. The efficiency and inefficiency of the packet time-slot duration are useful indicators of the hyperplane's operation. They are given in Eqns. 4.3 and 4.4.

$$\text{efficiency} = \frac{\left\lceil \frac{Pe}{Z} \right\rceil}{\left\lceil \frac{Pe}{Z} \right\rceil + (N-1)} \tag{4.3}$$

$$\text{inefficiency} = \frac{N-1}{\left\lceil \frac{P_e}{Z} \right\rceil + (N-1)} \tag{4.4}$$

It should be noted that the analytic models in [25, 26, 27, 28] have been modified to reflect the June 1995 smart pixel array design. The net effect of the new design is that the propagation delay is "programmable" and the throughput can approach 1. The reader is referred to [25, 26, 27, 28] for the final analytic models. (These papers are in the journal submission process which can take a few years to complete.) However, the analyses presented in this thesis are concerned solely with the June 1994 smart pixel array design.

4.2.2 Embeddings in the Dual-Stream Circular Hyperplane

The circular hyperplane, with two counter-rotating rings of Z optical bit-channels each, can support three different embedding schemes [26] as described in Chapter 2. Each embedding affects the packet time-slot duration as outlined below.

Optimized Edge Bandwidth Embedding

The edge bandwidth can be maximized, i.e., doubled, if half the edges of a network are embedded in one ring while the other half are embedded in the second ring. The transmission delay is then halved and the packet time slot duration becomes

$$S = \frac{\left\lceil \frac{P_e}{2Z} \right\rceil + (N-1)}{B} \text{ seconds} \tag{4.5}$$

Optimized Propagation Delay Embedding

Instead of maximizing the edge bandwidth, the propagation delay can be minimized, i.e, almost halved. With two counter rotating rings, when all the edges are embedded in both rings, the distance between furthest nodes is only halfway around the appropriate ring. The packet time slot duration becomes

$$S = \frac{\left\lceil \frac{P_e}{Z} \right\rceil + (\frac{N}{2}-1)}{B} \text{ seconds} \tag{4.6}$$

Optimized Edge Bandwidth and Propagation Delay Embedding

In the case of the networks analyzed here it is possible to realize an optimal edge bandwidth and propagation delay embedding at once.

$$S = \frac{\left\lceil \frac{P_c}{2Z} \right\rceil + (\frac{N}{2} - 1)}{B} \text{ seconds} \tag{4.7}$$

4.3 Probability of Packet Blocking and Acceptance

All of the networks analyzed are internally nonblocking. For any given node blocking or packet loss occurs whenever the number of packets arriving at a slice on a smart pixel array exceeds b, the number that can be removed from that slice [26]. Conversely, packet acceptance occurs when a packet is received at a smart pixel array and forwarded to the message processor on the node.

In this section, three distinct blocking probability models are presented; one for the single-stream circular and two for the linear hyperplane. The derivation for the circular hyperplane was presented in [26] and is reviewed here. The first derivation for the linear hyperplane is based on the sequential assignment of nodes to logical channels presented in [25]. After reviewing that derivation, the author presents a refinement where it is assumed that a node may no longer transmit a packet to itself. The author then presents a second derivation that relies on the novel interleaved assignment of nodes to logical channels. It is demonstrated that the probability of blocking is decreased for the Crossout and dilated Crossout switches. Having presented the probability analysis for the linear hyperplane, the requirements for accurately calculating probabilities of blocking and acceptance for the three embedding schemes in the dual-stream hyperplane are discussed.

All the derivations are for the dilated Crossout switch. Though the other networks have unique blocking probabilities, they are all special cases of the dilated Crossout switch.

4.3.1 Single-Stream Circular Hyperplane

Consider the embedding of an $N = 16, C = 4, K = 4$ Crossout in the single stream circular hyperplane shown in Fig. 4.1. Since there is only one stream, each node has one smart pixel array instead of two as is the case for linear and the dual-stream circular hyperplanes. As

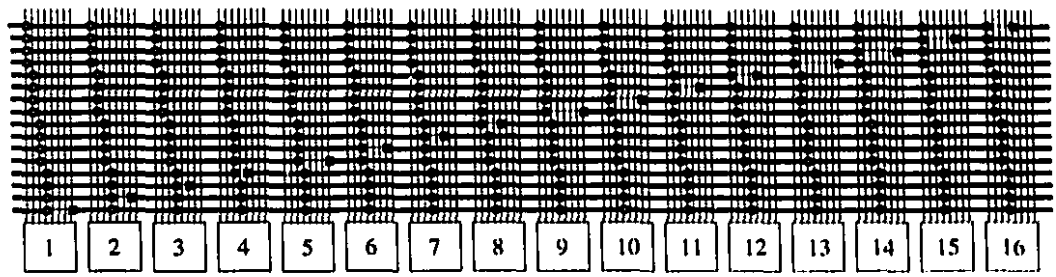


Figure 4.1: Embedding of an $N = 16, C = 4, K = 4$ Crossout in the single-stream circular hyperplane

a very important consequence, all the incoming packets to a node arrive on one array and are not distributed over two arrays. The derivations of the probabilities of blocking and acceptance are based on this embedding for which the following assumptions are made

- traffic is random and uniform
- all nodes are independent and statistically identical
- a node is equally likely to send a packet to any other node *including itself*

The probability that a node has a packet to send is α. The probability that packet is destined for a particular node is α/N. A slice can receive at most C packets at once. The probability that exactly j packets, in a particular combination, are destined for the slice is

$$P_c \equiv \left(\frac{\alpha}{N}\right)^j \left(1 - \frac{\alpha}{N}\right)^{C-j} \tag{4.8}$$

The probability that any j out of C packets are destined for a slice is given by the following binomial

$$P_j \equiv \binom{C}{j} \left(\frac{\alpha}{N}\right)^j \left(1 - \frac{\alpha}{N}\right)^{C-j} \tag{4.9}$$

The probability that $j \leq b$ packets arrive at a slice and can therefore be received without loss is

$$P_{rj} \equiv \sum_{j=1}^{b} \binom{C}{j} \left(\frac{\alpha}{N}\right)^j \left(1 - \frac{\alpha}{N}\right)^{C-j} \tag{4.10}$$

The probability that $j > b$ packets arrive at a slice and hence only b are received is

$$P_{rb} \equiv \sum_{j=b+1}^{C} \binom{C}{j} \left(\frac{\alpha}{N}\right)^j \left(1 - \frac{\alpha}{N}\right)^{C-j} \tag{4.11}$$

The expected number of packets that are received at a slice and sent to the message-processor is the sum of the expected number of packets, $j \leq b$, received without loss and the expected number of packets, $j > b$, received with loss and is given by

$$EP_{received} \equiv \sum_{j=1}^{b} j \binom{C}{j} \left(\frac{\alpha}{N}\right)^{j} \left(1 - \frac{\alpha}{N}\right)^{C-j} + \sum_{j=b+1}^{C} b \binom{C}{j} \left(\frac{\alpha}{N}\right)^{j} \left(1 - \frac{\alpha}{N}\right)^{C-j} \quad (4.12)$$

The expected number of packets that are received at a slice and not sent to the message-processor are those arriving beyond b and is given by

$$EP_{lost} \equiv \sum_{j=b+1}^{C} (j-b) \binom{C}{j} \left(\frac{\alpha}{N}\right)^{j} \left(1 - \frac{\alpha}{N}\right)^{C-j} \quad (4.13)$$

The conditional acceptance probability of packets over all K slices at a node for the dilated Crossout switch is given in Eqn. 4.14.

$$PA = \frac{K}{a\alpha} EP_{received} \quad (4.14)$$

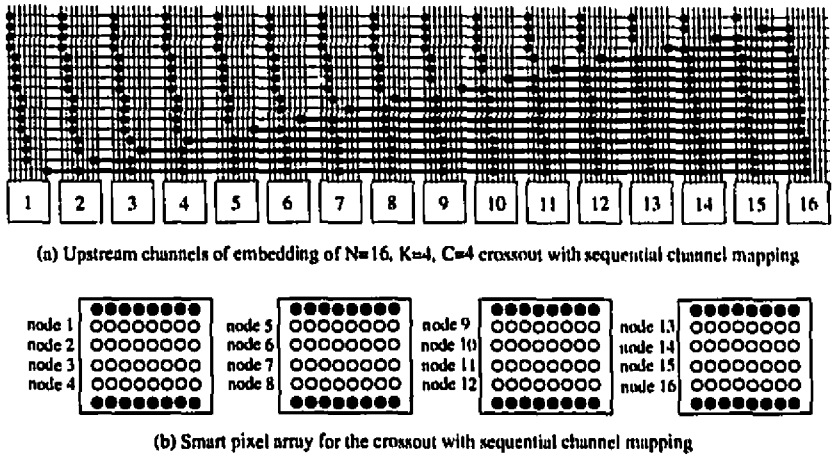
The conditional blocking probability can be calculated as 1-PA or directly from Eqn. 4.13 as follows.

$$PB = \frac{K}{a\alpha} EP_{lost} \quad (4.15)$$

4.3.2 Linear Hyperplane - Sequential Channel Assignment

Consider the embedding of an $N = 16, K = 4, C = 4$ Crossout switch in the linear hyperplane and its smart pixel array shown in Fig. 4.2a (only the upstream channels are shown). The second assumption, made for the single-stream circular hyperplane, which states that nodes are statistically identical, no longer holds. Consider the smart pixel array of node x in the upstream direction where $1 < x < N$. It can receive packets from nodes 1 to x but not nodes $x + 1$ to N. Thus, each node has a different number of incoming packets. In the single-stream circular case, this situation does not arise due to the closure of the rings; every node has the same number of incoming packets. The probabilities of blocking and acceptance must now be calculated for each node individually. An average over the N nodes will give the network's overall blocking and acceptance probability.

Assume that the nodes are assigned to logical channels sequentially as shown in Fig. 4.2b. Starting with the first node, its transmitters are mapped to the first logical channels of the



(a) Upstream channels of embedding of N=16, K=4, C=4 crossout with sequential channel mapping

(b) Smart pixel array for the crossout with sequential channel mapping

Figure 4.2: Sequential assignment of nodes to logical channels

first slice. The transmitters for the second node are mapped to adjacent logical channels on the same slice. When the slice is full, mapping proceeds to the next slice.[1] This assignment can also be seen in the embedding of Fig. 2.11a. Consider node four. The four incoming channels from nodes one to four are mapped to first slice while the other slices remain unused.

The probabilities can then be calculated as follows. Consider node x that may receive packets from nodes 1 to x. The number of possible incoming channels at node x is ax. Since each slice has C channels, the number of full slices is given by $\lfloor ax/C \rfloor$. If the number of incoming channels does not fit on an integral number of slices, there will be one slice that is partially full. The number of channels, W, on the partially full slice is given by

$$W \equiv ax - C\left\lfloor \frac{ax}{C} \right\rfloor \tag{4.16}$$

The expected number of packets received at a full slice is given in Eqn. 4.12. Similarly, the expected number of packets received at the partially full slice, ERP_x, is given by Eqn. 4.17.

$$\begin{aligned} ERP_x \equiv & \sum_{j=1}^{b} j \binom{W}{j} \left(\frac{\alpha}{N}\right)^j \left(1-\frac{\alpha}{N}\right)^{W-j} + \\ & \sum_{j=b+1}^{W} b \binom{W}{j} \left(\frac{\alpha}{N}\right)^j \left(1-\frac{\alpha}{N}\right)^{W-j} \end{aligned} \tag{4.17}$$

[1]The ordering of slices is actually irrelevant; the sequential assignment scheme requires only that transmitters from neighbouring nodes are mapped to the same slice until full.

The expected number of packets received at node x, in the upstream, is the number of full slices times the expected number of packets received at a full slice plus the expected number of packets received at the partially full slice and is given by

$$ER_x \equiv \left\lfloor \frac{ax}{C} \right\rfloor EP_{received} + ERP_x \tag{4.18}$$

The expected number of packets received in the downstream is derived similarly. It is assumed that a node may transmit a packet to itself in the upstream but not in the downstream. The expected number of packets received at node x over both streams is given by $ER_x + ER_{N-x}$. Hence the probability of acceptance over the N nodes at a normalized offered load of α is given by Eqn. 4.19.

$$PA_s = \frac{1}{a\alpha N}\left(2\sum_{x=1}^{N} ER_x - ER_N\right) \tag{4.19}$$

The blocking probability can be calculated simply as 1-PA, or directly as follows. The expected number of packets lost at a full slice is given by Eqn. 4.13. Similarly, the expected number of packets lost at a partially full slice is given in Eqn. 4.20.

$$ELP_x \equiv \sum_{j=b+1}^{W} (j-b)\binom{W}{j}\left(\frac{\alpha}{N}\right)^j\left(1-\frac{\alpha}{N}\right)^{W-j} \tag{4.20}$$

The expected number of packets lost at node x is the number of full slices times the expected number of packets lost at a full slice plus the expected number of packets lost at the partially full slice and is given by

$$EL_x \equiv \left\lfloor \frac{ax}{C} \right\rfloor EP_{lost} + ELP_x \tag{4.21}$$

Thus, the probability of blocking, PB_s, is given by

$$PB_s = \frac{1}{a\alpha N}\left(2\sum_{x=1}^{N} EL_x - EL_N\right) \tag{4.22}$$

The author now presents a refinement to the above model. The third assumption, which states that a node may transmit a packet to itself, is changed - it is now assumed that a node may not transmit to itself. This refinement brings the model closer to what an actual implementation would be. The α/N terms in the previous equations are changed to $\alpha/(N-1)$ since a node may transmit to only $N-1$ other nodes rather than N. In the upstream, node 1 will no longer receive any packets. Nodes 2 to N, will then receive

packets from nodes 1 to $N-1$. ER_x, given in Eqn. 4.18, which defined the expected number of packets received at node x can now be thought of as defining the expected number of packets received at node $x+1$. The downstream is identical. Therefore, the probability of acceptance and blocking are given by

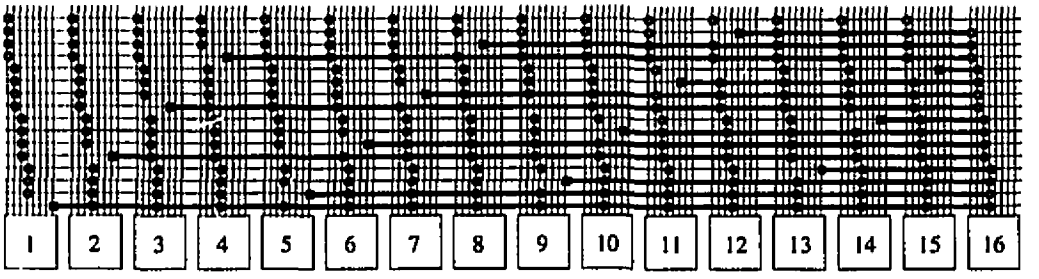
$$\begin{aligned} PA = \frac{2}{a\alpha N}\sum_{x=1}^{N-1}\Bigg[\left\lfloor\frac{ax}{C}\right\rfloor\Bigg(\sum_{j=1}^{b} j\binom{C}{j}\left(\frac{\alpha}{N-1}\right)^{j}\left(1-\frac{\alpha}{N-1}\right)^{C-j} + \\ b\sum_{j=b+1}^{C}\binom{C}{j}\left(\frac{\alpha}{N-1}\right)^{j}\left(1-\frac{\alpha}{N-1}\right)^{C-j}\Bigg) + \\ \sum_{j=1}^{b} j\binom{W}{j}\left(\frac{\alpha}{N-1}\right)^{j}\left(1-\frac{\alpha}{N-1}\right)^{W-j} + \\ b\sum_{j=b+1}^{W}\binom{W}{j}\left(\frac{\alpha}{N-1}\right)^{j}\left(1-\frac{\alpha}{N-1}\right)^{W-j}\Bigg] \end{aligned} \tag{4.23}$$

$$\begin{aligned} PB = \frac{2}{\alpha N}\sum_{x=1}^{N-1} \\ \Bigg[\left\lfloor\frac{ax}{C}\right\rfloor\Bigg(\sum_{j=b+1}^{C}(j-b)\binom{C}{j}\left(\frac{\alpha}{N-1}\right)^{j}\left(1-\frac{\alpha}{N-1}\right)^{C-j}\Bigg) + \\ \sum_{j=b+1}^{W}(j-b)\binom{W}{j}\left(\frac{\alpha}{N-1}\right)^{j}\left(1-\frac{\alpha}{N-1}\right)^{W-j}\Bigg] \end{aligned} \tag{4.24}$$

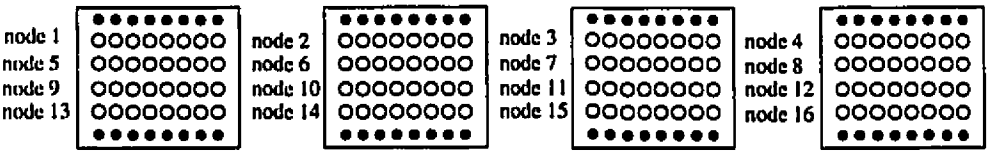
4.3.3 Linear Hyperplane - Interleaved Channel Assignment

The author now presents a novel embedding scheme that relies on an interleaved channel assignment scheme. There is an inefficiency in assigning nodes to channels in a sequential manner. Consider a smart pixel array for an $N = 16$ node network that has $K = 4$ slices and $C = 4$ channels per slice. As explained above node five would see its first slice filled. The blocking experienced on the first slice would be equivalent to that of a 4x4 Crossbar. The other three slices have no incoming channels and consequently their receivers are unused which is an under utilization of resources.

This premature filling of slices can be avoided if an interleaved channel assignment scheme is followed. The embedding of an $N = 16, K = 4, C = 4$ Crossout switch with an interleaved channel assignment is shown in Fig. 4.3a (only the upstream channels are



(a) Upstream channels of embedding of N=16, K=4, C=4 crossout with interleaved channel mapping



(b) Smart pixel array for the crossout with interleaved channel mapping

Figure 4.3: Interleaved assignment of nodes to logical channels

shown). This embedding should be contrasted with the embedding shown in Fig. 4.2a of the same Crossout switch with a sequential channel assignment. The transmitters from neighbouring nodes are mapped to channels on adjacent slices as shown for the smart pixel array in Fig. 4.3b. In the interleaved example above, the fifth node has one incoming channel on each of the four slices. Since there are four slices, blocking will be altogether avoided until node six, where the first slice has two incoming channels, from nodes one and five on the first slice (assuming that there is only one receiver per slice).

Consider node $x+1$, where $1 \leq x < N$. Of the slices on its smart pixel array, some of them will have $W_1 \equiv \lfloor ax/K \rfloor$ incoming channels while the others will have $W_2 \equiv \lceil ax/K \rceil$ incoming channels. The number of slices with W_1 incoming channels is

$$K_1 \equiv \begin{cases} ax - K * \lfloor ax/K \rfloor & \text{if } ax > K \\ 0 & \text{if } ax \leq K \end{cases} \tag{4.25}$$

The number of slices with W_2 incoming channels is

$$K_2 \equiv \begin{cases} K - (ax - K * \lfloor ax/K \rfloor) & \text{if } ax \geq K \\ 0 & \text{if } x < K \end{cases} \tag{4.26}$$

The interleaving scheme is optimal for the Crossout network, but is sub-optimal for the dilated Crossout switch where there are $a > 1$ transmitters per node. From the above

equations it can be seen that interleaving is implemented in groups of a transmitters from a node. In other words, the a transmitters of a node are mapped to a single slice. In order for the scheme to be optimal for the dilated Crossout switch, transmitters belonging to the same node, as well as those transmitters from adjacent nodes, should be mapped to channels of different slices.

The derivation of the expected number of packets on the slices with W_1 and W_2 incoming channels is analogous to that of the circular hyperplane's partially full slice. The expected number received on each one of the K_1 slices and similarly, on each one of the K_2 slices are given below.

$$EPR_{K_1} \equiv \sum_{j=1}^{b} j \binom{W_1}{j} \left(\frac{\alpha}{N-1}\right)^j \left(1-\frac{\alpha}{N-1}\right)^{W_1-j} + \sum_{j=b+1}^{W_1} b \binom{W_1}{j} \left(\frac{\alpha}{N-1}\right)^j \left(1-\frac{\alpha}{N-1}\right)^{W_1-j} \tag{4.27}$$

$$EPR_{K_2} \equiv \sum_{j=1}^{b} j \binom{W_2}{j} \left(\frac{\alpha}{N-1}\right)^j \left(1-\frac{\alpha}{N-1}\right)^{W_2-j} + \sum_{j=b+1}^{W_2} b \binom{W_2}{j} \left(\frac{\alpha}{N-1}\right)^j \left(1-\frac{\alpha}{N-1}\right)^{W_2-j} \tag{4.28}$$

The probability of packet acceptance is then

$$PA_i = \frac{2}{a\alpha N} \sum_{x=1}^{N-1} (K_1 \cdot E[PR_{K_1}] + K_2 \cdot E[PR_{K_2}]) \tag{4.29}$$

The expected number of packets lost at each one of the K_1 and K_2 slices are given by

$$EPL_{K_1} \equiv \sum_{j=b+1}^{W_1} (j-b) \binom{W_1}{j} \left(\frac{\alpha}{N-1}\right)^j \left(1-\frac{\alpha}{N-1}\right)^{W_1-j} \tag{4.30}$$

$$EPL_{K_2} \equiv \sum_{j=b+1}^{W_2} (j-b) \binom{W_2}{j} \left(\frac{\alpha}{N-1}\right)^j \left(1-\frac{\alpha}{N-1}\right)^{W_2-j} \tag{4.31}$$

The probability of packet blocking is then

$$PB_i = \frac{2}{a\alpha N} \sum_{x=1}^{N-1} (K_1 \cdot EPL_{K_1} + K_2 \cdot EPL_{K_2}) \tag{4.32}$$

Numerical Comparison between the Interleaved and Sequential Channel Assignment Schemes

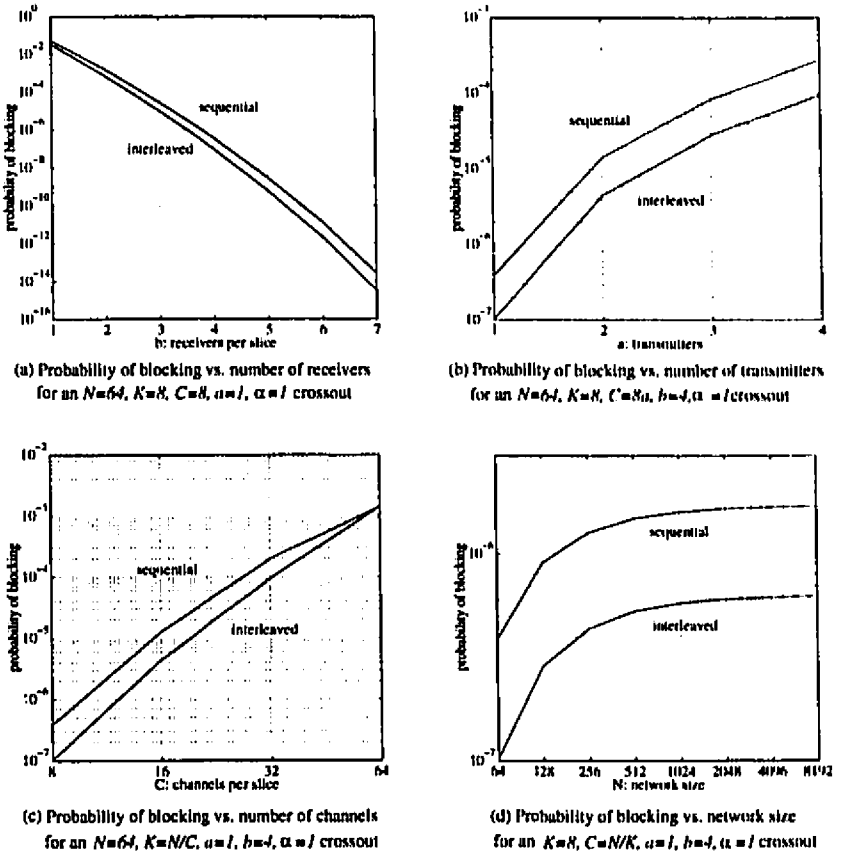
The decrease in blocking probability produced by the interleaved scheme with comparison to the sequential channel assignment scheme is shown in Fig. 4.4. The network is an $N = 64, K = 8, C = 8, a = 1, b = 4$ Crossout switch under full load ($\alpha = 1$). The four graphs examine the effect on blocking probability when varying b, a, C and N.

In Fig. 4.4a, the number of receivers per slice, b, is varied from one to seven. With seven receivers $PB_s = 2.82\text{x}10^{-14}$ and $PB_i = 3.53\text{x}10^{-15}$. The improvement is almost an order of magnitude. With an increasing number of receivers per slice, the interleaved scheme shows greater improvement over the sequential scheme. The improvement is due to the fact that the interleaved scheme will have no blocking until there are more than Kb incoming channels to a node whereas the sequential scheme will block as soon as there are more than b incoming channels. The reduction in blocking probability provided by interleaving is proportional to the number of receivers per slice when the number of transmitters is one. For example when $b = 5$, $PB_s/PB_i = 4.86$.

In Fig. 4.4b, the number of transmitters per node, a, is varied from one to four (the number of channels per slice is increased by a factor of a to compensate). For $a = 1$, $PB_s = 3.90\text{x}10^{-7}$ and $PB_i = 1.03\text{x}10^{-7}$ which shows a decrease in blocking by a factor of almost 4. As a increases there is a decrease in the advantage provided by interleaving. The decrease is due to that fact the granularity with which interleaving is done is increasing, i.e., in groups of a channels instead of a single channel at a time. If the interleaving was done with individual transmitters and not a simultaneously, the interleaved scheme would maintain the same advantage as the number of transmitters increased.

In Fig. 4.4c, the number of channels per slice, C, is varied between 8, 16, 32 and 64. As C increases, or conversely the number of slices decreases (given the fixed network size), the advantage of the interleaved scheme decreases. At the extreme when $C = N$, the Crossout degenerates into an $N\text{x}N$ Crossbar. When there is only one slice, both schemes produce identical results. This is expected since no interleaving is possible with a single slice.

In Fig. 4.4d, the network size, N, is varied from 64 to 8192 nodes. The number of channels is increased to compensate for the increase, i.e., the number of slices is kept



(a) Probability of blocking vs. number of receivers for an $N=64$, $K=8$, $C=8$, $a=1$, $\alpha=1$ crossout

(b) Probability of blocking vs. number of transmitters for an $N=64$, $K=8$, $C=8a$, $b=4$, $\alpha=1$ crossout

(c) Probability of blocking vs. number of channels for an $N=64$, $K=N/C$, $a=1$, $b=4$, $\alpha=1$ crossout

(d) Probability of blocking vs. network size for an $K=8$, $C=N/K$, $a=1$, $b=4$, $\alpha=1$ crossout

Figure 4.4: Blocking probability for interleaved versus sequential channel assignment

constant at 8. For $N = 64$ nodes, $PB_s = 3.90\text{x}10^{-7}$ and $PB_i = 1.03\text{x}10^{-7}$; for $N = 8192$ $PB_s = 1.70\text{x}10^{-6}$ and $PB_i = 6.25\text{x}10^{-7}$. Thus, as the network size increases, the advantage of the interleaved scheme over the sequential scheme decreases and does so to a constant factor. This decrease is due to the fact that as the number of channels per slice increases and the number of slices remains constant, the relative 'amount' of interleaving decreases for the given network size. This can be inferred from Fig. 4.4c where the N/K ratio is increasing as C increases (and K decreases). The same is true here as the N/K ratio increases by virtue of increasing N rather than C. If on the other hand the number of channels per slice is fixed at 8 and the number of channels per slice increased with the network size, the interleaved scheme would maintain its advantage over the sequential scheme.

The interleaved scheme decreases the packet loss rate for networks. This decrease is

due to the fact that the loss rate is directly proportional to the probability of blocking, PB, as will be shown in Section 4. However, the benefit to the aggregate bandwidth is minimal. The bandwidth is directly proportional to the probability of acceptance, PA, as will be shown in Section 4. Consider the following. Suppose that a reduction in PB from 10^{-4} to 10^{-5} is achieved with the interleaved scheme for a given Crossout switch. The corresponding increase in PA will be from 0.9999 to 0.99999 and so, represents an increase of only 0.009% for the aggregate bandwidth.

4.3.4 Dual Stream Circular Hyperplane

Each node in the dual-stream hyperplane has two smart pixel arrays per node as opposed to one for the single-stream hyperplane. Packet arrivals from all the other nodes in the backplane are not uniformly and identically distributed over both arrays similar to the linear hyperplane. Consequently the nodes are no longer statistically independent as was assumed for the single-stream circular hyperplane.

Maximized edge bandwidth

Consider the minimized propagation delay embedding of the Crossout in the dual stream circular hyperplane shown in Fig. 4.5. It can be seen that the probabilities of blocking and acceptance derived for the single-stream hyperplane do not apply. The second assumption being invalid as is the case for the linear hyperplane. In either stream every node cannot receive packets from any other node. Thus, all nodes are no longer statistically identical.

Consider the smart pixel array of node 7 in the upstream direction. It may receive packets from nodes 1 to 8 but not from nodes 9 to 16. Despite the fact that half of the nodes in the upstream are no longer transmitting, all of the slices on the arrays of each node are full and not half empty. They remain full because each node from 1 to 8 now transmits over two logical channels at once instead of one. This doubling of edge bandwidth or channel width results in effectively halving the number of logical channels on a slice from $C = 4$ to $C = 2$.

If it is assumed that a node may transmit to itself the calculation of the blocking probability would require that $C = 2$ instead of $C = 4$ and all nodes could be considered

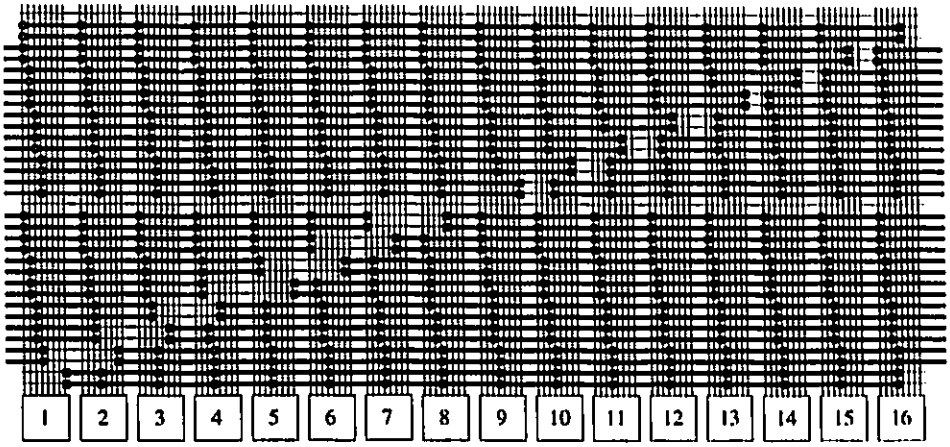


Figure 4.5: Maximized edge bandwidth embedding of the $N = 16, K = 4, C = 4$ Crossout switch in the dual-stream circular hyperplane

identical. If the assumption is made that a node may not transmit a packet to itself, two situations would have to be considered for nodes in either stream. Consider node i, where $1 \leq i \leq 8$, in the upstream. Its smart pixel array will see three out of four slices full but the last slice only partially full since the node does not transmit to itself. Consider node j, where $9 \leq j \leq 16$, in the upstream. Its smart pixel array will see all of its slices filled. Thus an exact calculation of the probabilities would have to take these factors into account.

Minimized propagation delay

Consider the minimized propagation delay embedding of the Crossout in the dual stream circular hyperplane shown in Fig. 4.6. As with the maximized edge bandwidth embedding, the second assumption (that all nodes are statistically independent) made for the single stream circular hyperplane does not hold. The nodes in either stream cannot receive packets from all other nodes. The nodes are then no longer statistically identical.

Consider the smart pixel array of node four in the upstream direction. It may receive packets from nodes 1 to 3 and nodes 12 to 16. The packets arrive on the first three channels of the first slice, the last channel of the third slice and all four channels on the last slice. In general, half of the channels will not have any incoming packets while the other half will cover a contiguous group of channels over several slices. The situation is similar for the

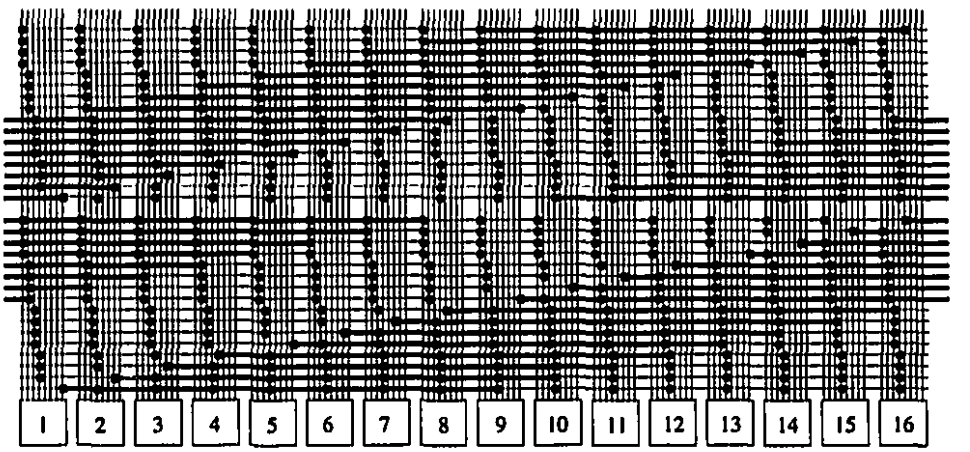


Figure 4.6: Minimized propagation delay embedding of the $N = 16, K = 4, C = 4$ Crossout switch in the dual-stream circular hyperplane

downstream direction.

The embedding shown can be considered to have a sequential assignment of transmitters of nodes to logical channels. This embedding could be optimized with an interleaved assignment as presented earlier. In either situation, an exact calculation of blocking probability would require careful consideration of the pattern of arriving packets onto the slices of the smart pixel array for each node.

Optimized propagation delay and edge bandwidth

Consider the optimized edge bandwidth and propagation delay embedding of the Crossout in the dual-stream circular hyperplane shown in Fig. 4.7. As with the previous two embeddings, the second assumption made for the single-stream circular hyperplane does not hold. The nodes in either stream cannot receive packets from all other nodes and are thus no longer statistically identical.

The doubled edge bandwidth causes the effective number of channels per slice to be halved as was the case for the maximized edge bandwidth embedding. Furthermore, each node in the upstream cannot transmit to any other node, but only to those within a distance of $N/2$. Thus the probability that a node has a packet destined for another node in a stream is $\alpha/(N/2)$. The analysis needed for an accurate calculation of blocking probability would

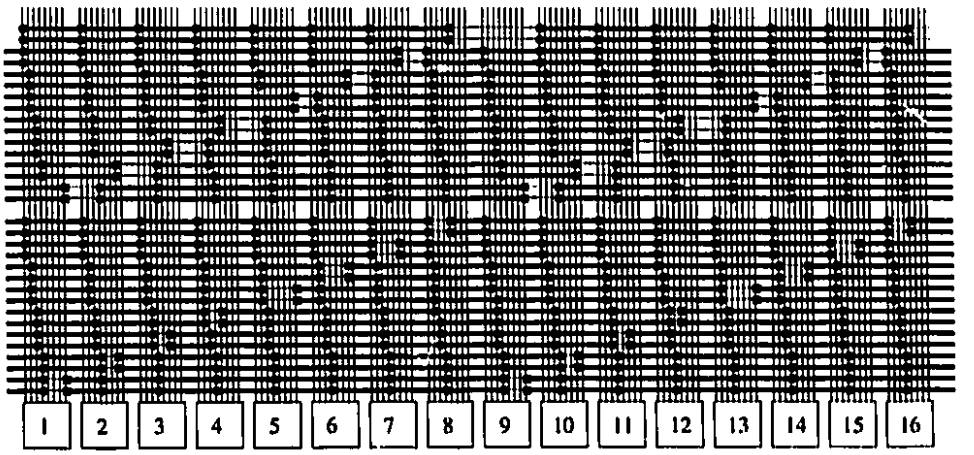


Figure 4.7: Maximized edge bandwidth and minimized propagation delay embedding of the $N = 16, K = 4, C = 4$ Crossout switch in the dual-stream circular hyperplane

have to take these factors into account.

4.3.5 Poisson Model of the Binomial

The binomial model for the probability of acceptance and blocking for the dilated Crossout switch tends to be somewhat computationally intensive, especially for embeddings in the linear hyperplane. The binomial can be approximated using a Poisson model which provides sufficient accuracy for a first order model [26]. The following well known theorem is quoted.

Theorem 4.1 *A binomial distribution with parameters n, p converges to the Poisson distribution with parameter λ if $n \to \infty$ and $p \to 0$ s.t. $np = \lambda$ is constant.*

Proof

$$\begin{aligned} P[X = k] &= \binom{n}{k} p^k (1-p)^{n-k} \qquad (4.33) \\ &= \frac{n!}{(n-k)!k!} p^k (1-p)n - k \\ &\cong \frac{n^k p^k (1-p)^{n-k}}{k!} \\ &\cong \frac{(\frac{\lambda}{n})^k n^k (1-\frac{\lambda}{n})^{n-k}}{k!(1-\frac{\lambda}{n})^k} \end{aligned}$$

$$\cong \frac{\lambda^k e^{-\lambda}}{k!}$$
$$\cong \frac{(np)^k e^{-np}}{k!}$$

Single Stream Circular Hyperplane

For the statistically independent nodes in the circular hyperplane the Poisson approximations of Eqn. 4.14 and Eqn. 4.15 yield

$$PA = \frac{K}{a\alpha}\left[\sum_{j=1}^{b} j\frac{e^{-\alpha C/N}(\alpha C/N)^j}{j!} + b\sum_{j=b+1}^{\infty}\frac{e^{-\alpha C/N}(\alpha C/N)^j}{j!}\right] \tag{4.34}$$

$$PB = \frac{K}{a\alpha}\sum_{j=b+1}^{\infty}(j-b)\frac{e^{-\alpha C/N}(\alpha C/N)^j}{j!} \tag{4.35}$$

Linear Hyperplane - Sequential Assignment

For the linear hyperplane with a sequential assignment scheme, the Poisson approximations of Eqn. 4.23 and Eqn. 4.24 yield

$$PA = \frac{2}{a\alpha N}\sum_{x=1}^{N-1}\left[\left\lfloor\frac{ax}{C}\right\rfloor\right. \tag{4.36}$$
$$\left(\sum_{j=1}^{b} j\frac{e^{-\alpha C/(N-1)}(\alpha C/(N-1))^j}{j!} + b\sum_{j=b+1}^{\infty}\frac{e^{-\alpha C/(N-1)}(\alpha C/(N-1))^j}{j!}\right) +$$
$$\left.\left(\sum_{j=1}^{b} j\frac{e^{-\alpha W/(N-1)}(\alpha W/(N-1))^j}{j!} + b\sum_{j=b+1}^{\infty}\frac{e^{-\alpha W/(N-1)}(\alpha W/(N-1))^j}{j!}\right)\right]$$

$$PB = \frac{2}{a\alpha N}\sum_{x=1}^{N-1}\left[\left\lfloor\frac{ax}{C}\right\rfloor\sum_{j=b+1}^{\infty}(j-b)\frac{e^{-\alpha C/(N-1)}(\alpha C/(N-1))^j}{j!} +\right.$$
$$\left.\sum_{j=b+1}^{\infty}(j-b)\frac{e^{-\alpha W/(N-1)}(\alpha W/(N-1))^j}{j!}\right] \tag{4.37}$$

Linear Hyperplane - Interleaved Assignment

For the linear hyperplane with an interleaved assignment scheme the Poisson approximation of Eqn. 4.29 and Eqn. 4.32 yield

$$PA = \frac{2}{a\alpha N}\sum_{x=1}^{N-1} \left[K_1\left(\sum_{j=1}^{b} j\frac{e^{-\alpha W_1/(N-1)}(\alpha W_1/(N-1))^j}{j!} + b\sum_{j=b+1}^{\infty}\frac{e^{-\alpha W_1/(N-1)}(\alpha W_1/(N-1))^j}{j!}\right) + K_2\left(\sum_{j=1}^{b} j\frac{e^{-\alpha W_2/(N-1)}(\alpha W_2/(N-1))^j}{j!} + b\sum_{j=b+1}^{\infty}\frac{e^{-\alpha W_2/(N-1)}(\alpha W_2/(N-1))^j}{j!}\right)\right] \quad (4.38)$$

$$PB = \frac{2}{a\alpha N}\sum_{x=1}^{N-1}\left[K_1\sum_{j=b+1}^{\infty}(j-b)\frac{e^{-\alpha W_1/(N-1)}(\alpha W_1/(N-1))^j}{j!} + K_2\sum_{j=b+1}^{\infty}(j-b)\frac{e^{-\alpha W_2/(N-1}(\alpha W_2/(N-1))^j}{j!}\right] \quad (4.39)$$

4.4 Network Throughput

4.4.1 Linear Hyperplane

The hyperplane is clocked at a rate of B bit-clock cycles per second. During one period, a bit on a single optical bit-channel can be transferred from one smart pixel array to an immediately neighbouring array. By operating Z bit-channels in parallel, a peak theoretical bandwidth of ZB bits/s exists in both the upstream and downstream directions. If the backplane is clocked at 1 Gbit/s and an array of 1024 optical bit-channels is used then a capacity of 1 Tbit/s is realized.

As mentioned previously, the analyses in [25, 26, 27, 28] have been revised to model the June 1995 smart pixel arrays, which allow a variable propagation delay. The variable nature of the propagation delay allows the throughput to approach unity. The reader is referred to [25, 26, 27, 28] for the revised analysis. The reader is reminded that this thesis concerns itself solely with the June 1994 smart pixel array design.

For any given network that is embedded in the synchronous hyperplane, time is divided into packet time-slot durations. During each time-slot a packet can be transferred from one

node to any other node in the network. For a single node with a transmitters, the number of bits per clock cycle that are transferred in a time-slot is given by aP/S where P has units of bits/packet and S seconds/packet. For all N nodes operating at a bit rate of B, this then becomes aNP/S. With contention at the output ports, some fraction with probability PB will not be delivered. The probability of acceptance derived in the previous section is conditional, i.e, it assumes that a node has a packet to transmit. The unconditional probability is found by multiplying by α. The complete expression for aggregate bandwidth [25] is given by Eqn. 4.40. As the architecture is scalable and able to accommodate large numbers of nodes in the backplane a more relevant measure of throughput is the bandwidth available to individual nodes, BW_n, given in Eqn. 4.41.

$$BW_a = \frac{\alpha a N(PA)P}{S} \text{bits/s} \tag{4.40}$$

$$BW_n = \frac{\alpha a(PA)P}{S} \text{bits/s} \tag{4.41}$$

While BW_a and BW_n are measures of actual carried traffic, they do not necessarily indicate of the peak capacities available which are useful measures and the author identifies as follows. The peak edge bandwidth, BW_e, which is defined as the maximum throughput supported per edge, is given by Eqn. 4.42. It is a measure of the maximum available bandwidth for each transmitter on any given node. Each of the networks has a maximum possible throughput based on their embedding. The peak capacity, BW_c, for a network is defined as the aggregate bandwidth at full load in the absence of blocking. It is given in Eqn. 4.43.

$$BW_e = \frac{ZB}{e} \text{bits/s} \tag{4.42}$$

$$BW_c = \frac{aNP}{S} \text{bits/s} \tag{4.43}$$

Complementing the measures of bandwidth are the measures of loss rates. The loss rate, BW_{loss}, is defined in Eqn. 4.44. It is identical to the equation for aggregate bandwidth except that PA has been replaced by PB. The sum of the loss rate and the aggregate bandwidth does not equal the capacity of the backplane (as will be explained later). The unused capacity, BW_{unused} is defined as the peak available bandwidth of the backplane less the peak capacity at a given offered load.

$$BW_{loss} = \frac{\alpha a N(PB)P}{S} \text{bits/s} \tag{4.44}$$

$$BW_{unused} = \alpha \left(ZB - \frac{aNP}{S} \right) \text{ bits/s} \tag{4.45}$$

4.4.2 Dual-Stream Circular Hyperplane

There are two embeddings in the dual-stream circular hyperplane that maximize the edge bandwidth. Half of the e edges of a given network are embedded into the Z optical bit-channels in the downstream ring while the other half are embedded into the other ring. Thus the e edges share $2Z$ channels unlike the linear hyperplane where e edges share Z bit-channels, in both streams. Thus the aggregate bandwidth for the maximized edge bandwidth schemes is considered to be $2ZB$ bits/s.

4.5 Input Queuing Analysis

The nodes in the hyperplane can be buffered with the addition of FIFO input queues which the message processor has the task of controlling. Each queue is modeled as a continuous time $M \mid M \mid Y \mid \infty$ queuing system, with memoryless inter-arrival time distribution and memoryless service time distribution [25]. The arrivals are modeled by a Poisson process and the departures are assumed to be adequately modeled also by a Poisson process. Each queue can have $Y \geq 1$ servers and either an infinite or finite size. The packets arrive at the queues at a rate of λ packets per second defined in Eqn. 4.46. The maximum service rate is μ_{max} packets per second given in Eqn. 4.47. It is defined as the maximum rate at which the servers can inject packets, which will be accepted into the backplane, in packets per second.

$$\lambda = \frac{\alpha ZB}{PN} \text{ packets/s} \tag{4.46}$$

$$\mu = \frac{PA}{S_{network}} \text{ packets/s} \tag{4.47}$$

While the queues should be modeled as discrete-time queues, the continuous-time model suffices. The former requires iteration to arrive at a solution while the latter analysis yields closed form solutions [26].

There are a number of measures of importance in the queuing model. Among them are expected number of packets in the queue, expected packet delay, packet loss probability and packet loss rate. First the infinite queue will be analyzed then the finite size queue. It

is desired to solve for the state of the queue under equilibrium conditions, from which the aforementioned measures can be calculated.

4.5.1 $M \mid M \mid Y \mid \infty$ Queue with Y Servers

The analysis for the infinite size queue was presented in [3] and is quickly reviewed here. Let P_i denote the equilibrium probability that the queue has i packets. Under the equilibrium conditions, the forward rate of flow must be equal to be backward rate of flow. The rate of a forward transition from state P_i to state P_{i+1} is always equal to the arrival rate λ. The rate of backward transitions depends on the state from which the transition takes place. For states P_i where $0 \leq i \leq Y$ the rate is $i\mu$. A state with only i packets in the queue can only lose packets at a maximum rate of $i\mu$ despite the existence of $Y > i$ servers. For the remaining states $Y < i$ the rate is $Y\mu$. An illustration of the Markov chain is shown in Fig. 4.8.

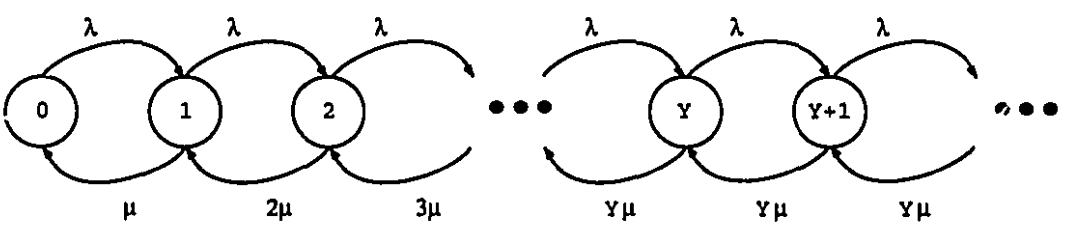


Figure 4.8: Markov chain model for an $M \mid M \mid Y \mid \infty$ queue

Since the flow balance property must hold (under equilibrium) between any two states, the following can be written.

$$\begin{aligned} \lambda P_0 &= 1\mu P_1 \\ &\vdots \\ \lambda P_{i-1} &= i\mu P_i \\ &\vdots \\ \lambda P_{Y-1} &= Y\mu P_Y \\ &\vdots \\ \lambda P_{j-1} &= Y\mu P_j \end{aligned} \tag{4.48}$$

$$\vdots$$

By rearranging to isolate P_i

$$P_i = P_0\left(\frac{\lambda}{\mu}\right)\frac{i}{i!} \qquad 0 \le i \le Y \tag{4.49}$$

$$P_i = P_0\left(\frac{\lambda}{\mu}\right)\frac{1}{Y!}\left(\frac{1}{Y}\right)^{i-Y} \qquad i > Y \tag{4.50}$$

By allowing $\rho = \lambda/Y\mu$

$$P_i = P_0\frac{(Y\rho)^i}{i!}\ 0 \le i \le Y \tag{4.51}$$

$$P_i = P_0\frac{Y^Y}{Y!}\rho^i\ i > Y \tag{4.52}$$

It remains that P_0 be found explicitly. Knowing that $\sum_{i=0}^{\infty} P_i = 1$ leads to

$$\sum_{i=0}^{Y} P_i + \sum_{i=Y+1}^{\infty} P_i = 1$$

$$P_0\sum_{i=0}^{Y}\frac{(Y\rho)^i}{i!} + P_0\frac{Y^Y}{Y!}\sum_{i=Y+1}^{\infty}\rho^i = 1$$

$$P_0 = \frac{1}{\sum_{i=0}^{Y}\frac{(Y\rho)^i}{i!} + \frac{(Y\rho)^Y}{Y!}\frac{\rho}{1-\rho}} \tag{4.53}$$

Eqn. 4.53 is mathematically equivalent to that found in Kleinrock [32]. Having found the probability of an empty queue explicitly, the probability that the queue has i packets is then also known. The expected number of packets, $Exp[C]$, in the queue is found by taking the expectation over the queue states.

$$Exp[C] = \sum_{i=0}^{Y} iP_i + \sum_{i=Y+1}^{\infty} iP_i \tag{4.54}$$

$$Exp[C] = P_0\sum_{i=0}^{Y} i\frac{(Y\rho)^i}{i!} + P_0\frac{Y^Y}{Y!}\sum_{i=Y+1}^{\infty} i\rho^i$$

$$Exp[C] = \frac{\sum_{i=0}^{Y}\frac{(Y\rho)^i}{i!} + \frac{(Y\rho)^Y}{Y!}\frac{\rho}{1-\rho}\left(\frac{1}{1-\rho} + Y\right)}{\sum_{i=0}^{Y}\frac{(Y\rho)^i}{i!} + \frac{(Y\rho)^Y}{Y!}\frac{\rho}{1-\rho}} \tag{4.55}$$

The expected packet delay, Exp[D], of a packet through the queue can be found using Little's Law [29]. It states that the expected delay is equal to the expected number of packets in the queue over the throughput. For an infinite size queue there are two discrete cases of throughput. If the arrival rate, λ, is less than the aggregate service rate, $Y\mu$, the throughput is λ. Packets cannot leave the queue at a greater rate than at which they arrive. If the arrival rate is greater that the aggregate service rate, the throughput is $Y\mu$. Packets cannot leave the queue at a greater rate than can be serviced. The expected delay is then

$$Exp[D] = \frac{Exp[C]}{\lambda} \quad \text{if } \lambda \leq Y\mu \tag{4.56}$$

$$Exp[D] = \frac{Exp[C]}{Y\mu} \quad \text{if } \lambda > Y\mu \tag{4.57}$$

Since the queue size is infinite packet loss cannot occur. Even if the arrival rate is greater than the aggregate service rate, packets are never dropped but rather added to the queue. In such situations, the number of packets in the queue grows indefinitely and consequently so do the expected packet delay and number of packets in the queue.

4.5.2 $M \mid M \mid Y \mid Q_s$ Queue with Y Servers

Having reviewed the analysis for the infinite size queue, the analysis for the finite size queue [31] is briefly reviewed. While the infinite queue model provides an indication of performance in the presence of input buffering it is a hypothetical one. A more realistic model is one using a queue of finite size, Q_s. The Markov chain model [32] is shown in Fig. 4.9. Note that $Q_s > Y$. Let P_i denote the equilibrium probability that the queue has

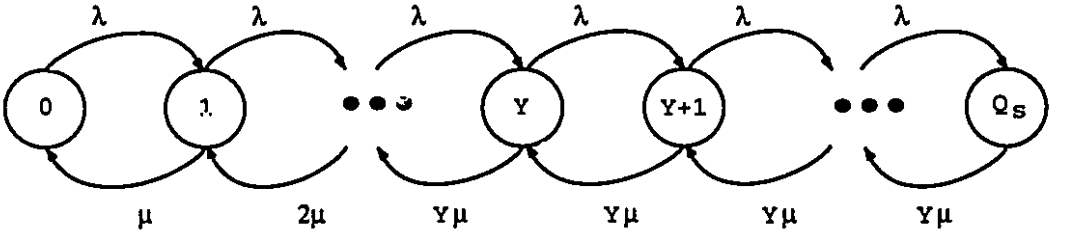


Figure 4.9: Markov chain model for an $M \mid M \mid Y \mid Q_s$ queue

i packets. Let λ denote the arrival rate of packets to the queue and μ is the departure rate from each server. Let $\rho = \lambda/(Y\mu)$.

$$\begin{aligned} \lambda P_0 &= 1\mu P_1 \\ \lambda P_1 &= 2\mu P_2 \\ &\vdots \\ \lambda P_{Y-1} &= Y\mu P_Y \\ \lambda P_Y &= Y\mu P_{Y+1} \\ &\vdots \\ \lambda P_{Q_s-1} &= Y\mu P_{Q_s} \end{aligned} \tag{4.58}$$

The probability of an empty queue is given by

$$P_0 = \frac{1}{\sum_{i=0}^{Y} \frac{(Y\rho)^i}{i!} + \frac{(Y\rho)^Y}{Y!}\rho\frac{1-\rho^{Q_s-Y}}{1-\rho}} \tag{4.59}$$

Having found the probability of an empty queue, the expected number of packets in the queue, $Exp[C]$, is given by

$$\begin{aligned} Exp[C] &= \sum_{i=0}^{Y} iP_i + \sum_{i=Y+1}^{Q_s} iP_i \\ Exp[C] &= P_0\left[\sum_{i=1}^{Y} i\frac{(N\rho)^i}{i!} + \frac{(Y\rho)^Y}{Y!}\rho\left(\rho\frac{1-(Q_s-Y)\rho^{Q_s-Y-1}+(Q_s-Y-1)\rho^{Q_s-Y}}{(1-\rho)^2}\right.\right. \\ &\quad \left.\left.(Y+1)\frac{1-\rho^{Q_s-Y}}{1-\rho}\right)\right] \end{aligned} \tag{4.60}$$

The expected packet delay, $Exp[D]$, is, again, found using Little's Law. For the finite size queue, unlike the infinite size queue, there is only one case for throughput, $Exp[TP]$. It is defined as the product of the arrival rate and the probability of the queue not being full.

$$Exp[TP] = \lambda(1 - P_{Q_s}) \tag{4.61}$$

The expected packet delay is then

$$Exp[D] = \frac{Exp[C]}{\lambda(1 - P_{Q_s})} \tag{4.62}$$

4.6 Internally Nonblocking Networks

The performance measures derived in the previous sections are general in nature, applying to the dilated Crossout switch. The Knockout switch, Fully Connected network, Crossbar, dilated Crossbar and Crossout switches can be considered as special cases of the dilated Crossout switch. The differences between these networks are detailed below [25, 26].

4.6.1 Crossbar

The hypergraph model of an NxN Crossbar has $e = N$ edges [33]. The Crossbar is a special case of the dilated Crossout switch where $(N, C, K, a, b) = (N, N, 1, 1, 1)$.

4.6.2 Knockout Switch

The hypergraph model of an NxNL Knockout switch has $e = N$ edges [33]. The Knockout switch is a special case of the dilated Crossout switch where $(N, C, K, a, b) = (N, N, 1, 1, L)$.

4.6.3 Fully Connected Network

An N node, Fully Connected network has $e = \frac{N(N-1)}{2}$ edges in both the upstream and downstream channels. Unlike the other networks considered here, there are no broadcast channels - all the edges are direct links from a transmitting node to a single destination node. The Fully Connected network is a special case of the dilated Crossout switch where $(N, C, K, a, b) = (N, N^2, 1, N, N)$. Its blocking probability is identically zero since every node can accept all $N - 1$ packets arriving simultaneously from the other $N - 1$ nodes. Thus, no packet loss due to blocking will ever occur.

The aggregate bandwidth of the Fully Connected network depends not only on the embedding but also the number of transmitters available on each node. In theory, each node could have a maximum of $N - 1$ transmitters, one for each outgoing edge. However, this would be impractical due to the hardware costs of implementing $N - 1$ transmitters per node and complexity in controlling them. Instead, a sub-optimal but practical number, $Y < N - 1$, transmitters per node are used.

4.6.4 Dilated Crossbar

An $aNxbN$ dilated Crossbar switch is very similar to the regular Crossbar. Its hypergraph model has $e = aN$ edges [33]. Each node has an input dilation allowing for the transmission of a packets simultaneously and an output dilation allowing for the reception of b packets simultaneously. The dilated Crossbar is a special case of the dilated Crossout where $(N, C, K, a, b) = (N, aN, 1, a, b)$.

4.6.5 Crossout Switch

The Crossout switch has nodes with input dilation of one. Like the Crossbar it has $e = N$ edges. The Crossout switch is a special of the dilated Crossout switch where $(N, C, K, a, b) = (N, C, K, 1, b)$.

4.7 Summary

In this chapter the general analytic model and various performance measures for the hyperplane were described. The packet time-slot duration, the probabilities of packet blocking in the single-stream circular and linear hyperplanes, the throughput and loss rates were described. Two input queuing models were then briefly reviewed. Finally, the specific instances of six internally non-blocking networks were discussed.

Chapter 5

Software Operation

5.1 Introduction

The analytic model of the hyperplane, described in the previous chapter, is implemented in software. The author presents a guide for operating the software to facilitate its future use. The tool is divided into two sections: (i) numerical analysis and (ii) numerical visualization. The first section provides the user with the tools to perform an analysis of different hyperplane architectures under a variety operating conditions. The latter section allows the user to view the output of the previous analysis and assess the results.

5.2 Numerical Analysis

The analysis portion of the tool is started by typing **analyze_hp** at the Matlab command line. In order to provide a simple method of performing a design space exploration, a fully automated, menu driven interface is presented. Upon startup an initial title window is displayed. Since there are a large number of parameters relating to the hardware architecture to be configured and operating conditions to be set, a series of menus are presented which guide the user through the choices. Once all the choices have been made, the analysis proceeds to compute the performance measures described in Chapter 4. Upon completion the results are saved into a single data file. They can be viewed using the visualization portion of the software tool described in the next section.

Hyperplane Architecture

Following the title window is the first menu, shown in Fig. 5.1. It presents two choices for the architecture of the hyperplane which are the linear and the dual-stream circular models. The selection is made by clicking on the desired hyperplane type. The single-stream hyperplane is not modeled as it does not have any advantages over the linear model.

The analysis of the linear hyperplane can be of two orders of magnitude slower than that for the circular version, due to the difference between the functions that calculate the probabilities of acceptance and blocking for the two hyperplanes.

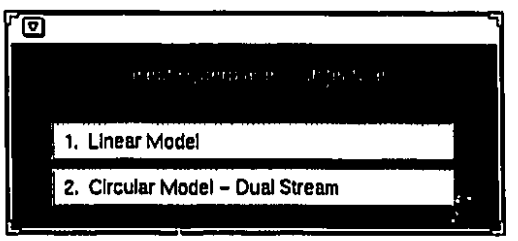


Figure 5.1: Hyperplane Architecture selection menu

Channel Assignment

If the linear model is chosen, a choice must be made regarding the node (or transmitter in the case of dilated networks) to logical channel assignment scheme. The two choices are the sequential assignment and the interleaved assignment (Fig. 5.2). The latter decreases the probability of blocking for the Crossout and dilated Crossout switches that have multiple independent slices. The channel assignment has no effect on the other networks.

Embedding Scheme

If the dual-stream circular model is chosen instead of the linear model, one of three embedding schemes, as discussed in Chapter 2, must be selected. The three choices are shown in Fig. 5.3. The first scheme maximizes (doubles) the edge bandwidth of the embedded network. The second scheme minimizes (approximately halves) the propagation delay and

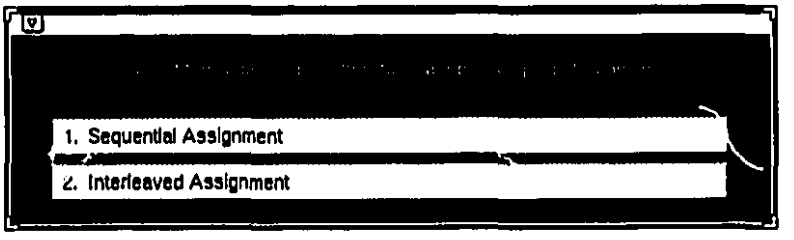


Figure 5.2: Channel Assignment selection menu

the third scheme achieves both simultaneously. The highest throughput is achieved with the third setting at no addition cost to hardware as compared to the first two schemes.

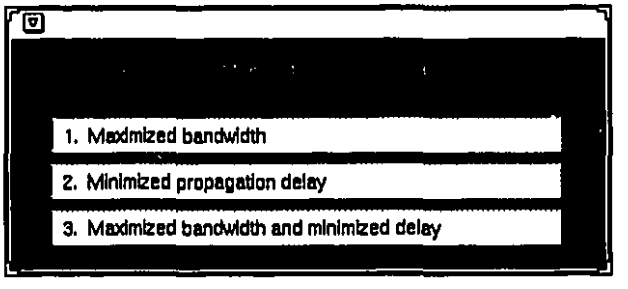


Figure 5.3: Embedding Scheme selection menu

Probability Model

There are three choices for the probability model as shown in Fig. 5.4. They represent a trade-off in speed versus accuracy. It is recommended that the second setting be used when very accurate, but not exact, results related to throughput and blocking probability are required. The accuracy is almost always identical to that of the exact model but is far less computationally intensive. The accuracy is maintained due to careful rounding off which, in some cases, results in some error in the 15th significant figure, but often produces no error whatsoever.

The third setting should be used whenever reasonably accurate results for all measures except for those that are directly proportional to the probability of blocking are required. The Poisson model is the quickest and produces very good results with regards to all the

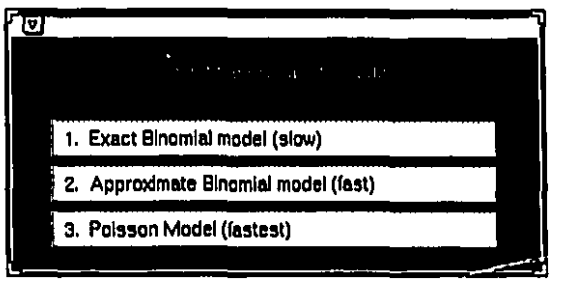


Figure 5.4: Probability Computation Model selection menu

measures that are not directly dependent on the probability of blocking. The probability of blocking may be incorrect by an order of magnitude when it is on the order of 10^{-5} or less. However the corresponding error in the probability of acceptance will be small, e.g., less than $10^{-3}\%$, since $PA = 1 - PB$.

In addition to the three settings, the probability model depends on the hyperplane architecture selected and the embedding scheme as described in Chapter 4. The appropriate model is automatically selected once the architecture is chosen. In the case of the dual-stream hyperplane, the probability model used is that of the single-stream hyperplane which assumes that every node receives packets from every other node. While this is not true due to the nature of the embeddings as described in Chapter 4, the accuracy is deemed sufficient for a first order analysis for design space exploration purposes.

Hardware Settings and Operating Conditions

Having decided upon the architecture, channel assignment or embedding scheme, and probability computation model, there now remain a number of hardware parameters and operating conditions to be set. The menu presented in Fig. 5.5 lists the seven parameters and shows their default settings. These can be changed to the desired values according to the particular exploration being done. Rudimentary checks are performed to ensure that the numbers entered are within their proper respective ranges.

1. The packet size, P, is the size in bits of the packets of data that are transmitted between nodes. The minimum size is eight bits. Should a value less that this be

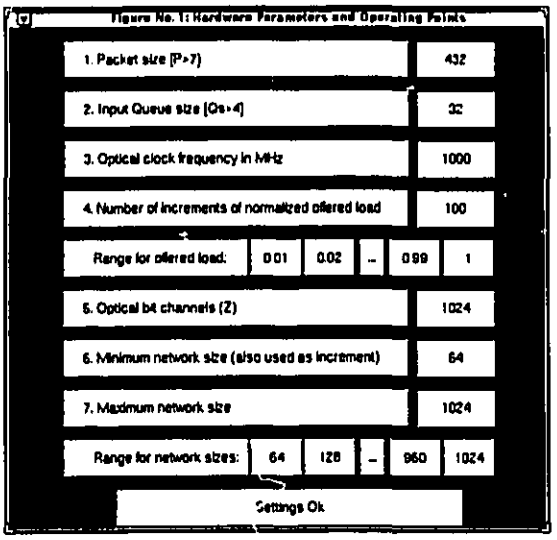


Figure 5.5: Hardware Settings and Operating Conditions window

entered, the setting will be forced to eight. The upper bounded is unlimited. The default setting is 432 bits; this was chosen as it is representative of a 53 byte ATM cell defined by the SONET standard [3] plus an 8 bit header.

A new value is entered by first clicking once on the number and then typing over the entry. In order for the update to take effect the mouse pointer should be moved off the window as typing Return does not work.

2. The queue size, Q_s, defines the number of packets an input queue can hold not including the one with the server. The size is lower bounded by 1. No upper limit is set. Should a value less than this be entered, the setting will be forced to four. The default setting is a queue size of 32 packets.

3. The clock rate, B of the backplane is specified in MHz. No bounds are enforced on the clock rate except that a zero is not accepted.

4. Analyses of the networks are performed over a range of offered loads. Since the offered load is normalized the range is always from zero to one. The user can specify the number of increments to be taken. The inverse of the number of increments is used as the initial load value as well as the step size. The default setting for the

number of increments of load is 100. This converts to a step size of 0.01 which is also used as the initial value in the range. (Using an initial value of identically zero would have necessitated the special handling in the queuing analysis equations.) The range over which the iterations are to be performed is shown in the line below the entry for load. If the user sets the number of increments to 40, for example, the range display would reflect the new setting by displaying the values of $0.025, 0.050, ..., 0.975, 1$.

The default of 100 was chosen since it provides for smooth curves when plotting the measures, such as the expected packet delay, which vary versus offered load. A second reason relates to the fact that for certain networks, infinite queues saturate or 'blow up' at very low values of offered load. If too few increments are selected, i.e., the step size is too large, the point at which the queue saturates may be bypassed. When plotting curves, the one for that particular network may not appear at all. The minimum number of increments is two; the networks would be analyzed at normalized offered loads of 0.5 and 1.

5. The number of free-space optical bit-channels Z, is lower-bounded by 16. Should a value less than this be entered, the setting will be forced to 16. No upper bound is placed on Z. The default setting is 1024. This value was chosen so that combined with a clock frequency of 1 GHz, the peak capacity of the hyperplane would be 1 Tbit/s.

   The number of optical bit-channels is an upper limit on the number of nodes that can be supported as each node (or transmitter in the case of dilated networks) must have least one optical bit-channel. Should Z be decreased to a value which is less than the current maximum network size, the maximum network size will be decreased to the largest integer multiple of the minimum network size, less than Z. Should be Z be decreased to a value that is lower than the minimum network size, it will be set to half of Z and the maximum network size will be set to Z. These adjustments are done automatically and the changes are reflected in the appropriate buttons.

6. Analysis of the networks is done not only over a range of offered loads but also over a range of network sizes, i.e., the number of nodes in a network. The minimum network size has a default setting of 64 nodes. This value is used as both the increment and

initial value for iterating over a range of network sizes.

The minimum network size is lower bounded by one and upper bounded by $Z/2$. The upper bound is to allow for a maximum network size equal to Z. The range is then from $Z/2$ to Z nodes.

It is recommended that the minimum network size be set to approximately one sixteenth the maximum network size. This size provides for reasonably smooth graphs when plotting curves versus network size and avoids excessive computation.

7. The maximum network size is the upper limit for the iterations over the range of nodes. The maximum network size must be an integer multiple of the minimum network size and be upper bounded by Z. The upper bound is set since every transmitter on each node must be assigned its own channel. [1] Network sizes entered that are not integer multiples of the minimum are rounded to the nearest multiple.

   The line below the entry for maximum network size displays the iterations over the range of network sizes. The starting value is the minimum network size and continues in increments of the minimum size until the maximum is reached. The display is automatically updated to reflect any changes in any of the associated parameters.

5.2.1 Embedded Networks

A few assumptions have been made for the networks, except for the Crossbar, described in Chapter 2 which are embedded in the hyperplanes. For each one of the networks, one or more parameters such as the input dilation, the number of slices, etc., has been fixed in the software to some arbitrary value. The settings are described below.

In order to change these values, the user will have to modify the code for the particular network in question. The routines are found in the */dse_tool/analysis/networks/* path. Each routine has the prefix *switch_* and is followed by the name of the network and a suffix of *.m*. The parameters are found at the beginning of each routine.

[1] This is not strictly true for certain embeddings where multiple transmitters use the same logical channels.

Dilated Crossbar

The input dilation, i.e., transmitters per node, has been set to 4. Input dilations $a > 1$ imply that the maximum network size that can be embedded is $N = Z/a$, since the total number of transmitters, aN, cannot exceed, Z, the number of optical bit-channels available. Thus the maximum network size that can be supported is reduced by a factor of a over the regular Crossbar. The software, however, does not model this limitation. The channel width is taken to be Z/aN which may be a fractional number. This can be corrected, if desired, by skipping the calculations when the limit of $N = Z/a$ has been reached.

The output dilation has been set to $b = 8$. For $N \leq 8$ there is no blocking since the number of packets destined for a node can never exceed the number of receivers.

Knockout switch

The Knockout switch has the same properties as an NxN Crossbar except for an N-to-L concentrator at each one of the N outputs. An output dilation of $L = 8$ has been set. For an infinite size Knockout switch, this produces a blocking probability on the order of 10^{-7} at full load.

Fully Connected network

The Fully Connected network is different from other networks in that it does not block given any traffic pattern. Thus no calls to routines are made to calculate probabilities; the acceptance probability is set to 1 and blocking to zero. The computation time needed is very short.

As with the dilated Crossbar, there is a limitation on the number of nodes which can be supported by the backplane. The Fully Connected network has $\frac{N(N-1)}{2}$ edges which must be less than Z since each edge must have its own logical channel. This limitation is not modeled by the software. For a hyperplane with $Z = 1024$ logical channels, the largest Fully Connected network that could be supported is 45 nodes which would require $\frac{N(N-1)}{2} = 990$ optical bit-channels.

The number of transmitters per node is set at 4 in the software. This could be increased

up to $N-1$ which would show an increase in the throughput but with greater hardware cost.

Crossout Switch

The Crossout switch can have several different configurations. The number of slices, K has been fixed to 8. The number of channels per slice is determined by the number of nodes in the network, i.e., $C = N/8$. The number of receivers per slice has been fixed to 4. When network sizes are small, i.e., 32 or less, the number of channels per slice will be less than or equal to 4, in which case the probability of blocking becomes identically zero.

Dilated Crossout Switch

The dilated Crossout switch is identical to the Crossout switch except that each node may have multiple transmitters. As with the dilated Crossbar, the maximum number of nodes, i.e., network size, permissible is Z/a. The input dilation has been set to 4. The number of slices have been set to 8, while the number of channels per slices is dependent on the network size, i.e., $C = aN/8$.

5.3 Numerical Visualization

In order to provide a simple method of visualizing the results of a design space exploration, a fully automated menu driven interface is presented. The visualization portion of the tool is started by typing **visualize_hp** at the Matlab command line. Upon startup an initial title window is displayed. The data from the last numerical analysis is loaded into Matlab's workspace.

Parameter Display

Following the title window, the parameters used in the design space exploration are displayed in the window shown in Fig. 5.6. The upper half of the window contains the choices made in the numerical analysis routine regarding the hyperplane architecture, channel assignment mapping or embedding scheme, and probability model. The lower half displays

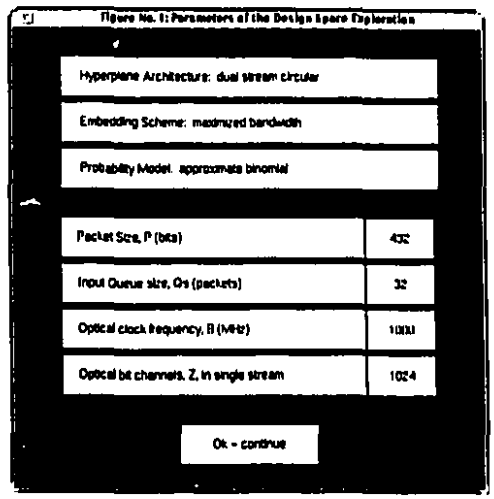


Figure 5.6: Parameters of the Design Space Exploration window

the packet size, input queue size, bit rate and the number of optical bit-channels in the single-stream.

Network Size and Offered Load Point

Following the parameter display window, the window shown in Fig. 5.7 appears. Most of the performance measures calculated vary with both network size and offered load, i.e., they could be plotted as three dimensional graphs. However, the graphing routine plots the performance measures as two dimensional graphs, i.e., against either network size or offered load. When graphing a given measure versus network size, the offered load at which the plot is generated must be fixed and vice versa.

The range of network sizes over which the analysis was performed is shown. By default the selected network size is the largest one. The user may change this to any network size within the range. Should a value which is lower than the minimum size be entered, the setting will be forced to the minimum network size. Similarly, if the value entered is above the range, the setting will be forced to the maximum network size. Any value entered that is within the range must be an integer multiple of the minimum network size. If the value is not a multiple, the setting will be rounded to the nearest multiple.

The number of points of offered load over which the analysis was performed is also

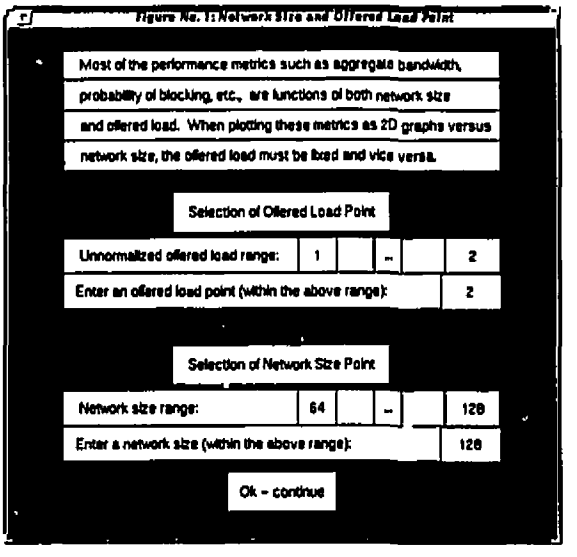


Figure 5.7: Network Size and Offered Load Point window

shown. By default the offered load point is set to the maximum or full load. The user may change this by entering any integer within the displayed range. Upper and lower bounds are enforced in a manner identical to that for the network size.

Curve Menu

Plotting of the individual curves is done through a menu shown in Fig. 5.8 which appears once the network size and offered load point have been set. There are a total of 21 performance curves. The user can select any one by clicking on the desired menu item. The menu is then closed and a graph window, shown in Fig. 5.9, is opened.

One curve is plotted for each of the six networks. The curves are plotted one at a time so that the user can place a label by clicking beside the associated curve. The network to which the current curve belongs to is shown in the command window along with its colour according to the list below.

1. Dilated Crossbar switch - yellow
2. Dilated Crossout switch - magenta

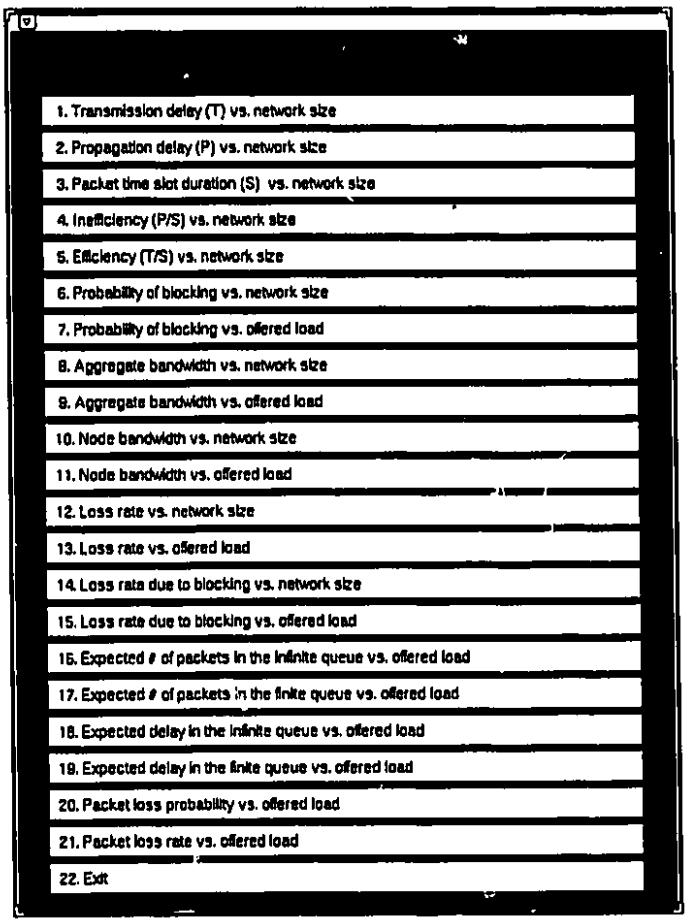


Figure 5.8: Curve Menu

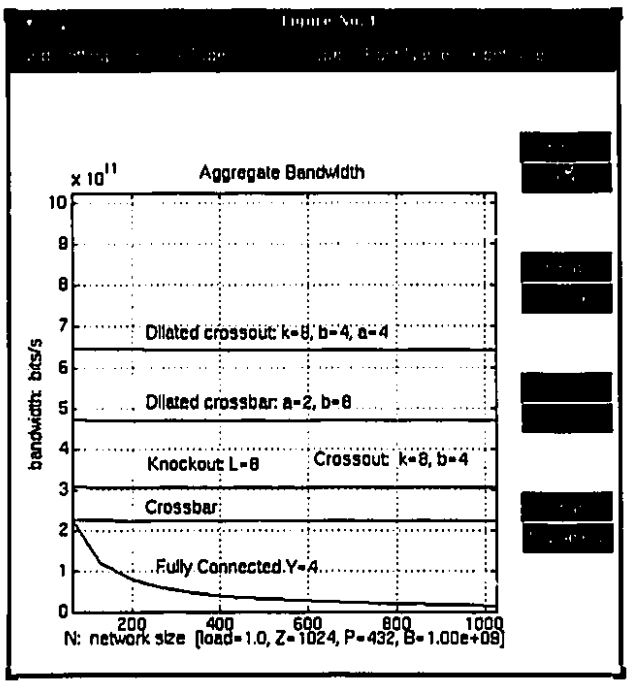


Figure 5.9: An output graph window

3. Knockout switch - red

4. Crossout switch - cyan

5. Crossbar - yellow

6. Fully Connected network - green

Once the six curves have been labeled, the user can use the tools to modify several of the graph and figure properties. The range of either x or y axes can be changed. A new value is entered by clicking on the current value of the axis and typing over the entry. For the new value to take effect the mouse must be moved off the graphics window.

There are five pulldown menus at the top of the figure window. The Grid Setting menu has two choices, On and Off, which allow the grid to be turned on or off. The X Axis Type and Y Axis Type menus have two entries - Linear and Log. The axes can be independently changed to achieve linear, semilogarithmic or log-log graphs. The Font Name and Font Size menus allow the type and size of the numbers on the axes to be altered.

Once the sixth curve of the graph has been labeled, the menu for selecting curves reappears. If a new performance measure is selected the currently displayed graph is erased and the new one plotted. Otherwise the graph can be closed.

5.4 Analysis Considerations

As mentioned in Chapter 3, the analysis is performed over a three dimensional space. The performance measures are plotted as two dimensional graphs versus either offered load or network size. These can be viewed as slices in the x-z and y-z planes, of the plot shown in Fig. 5.10.

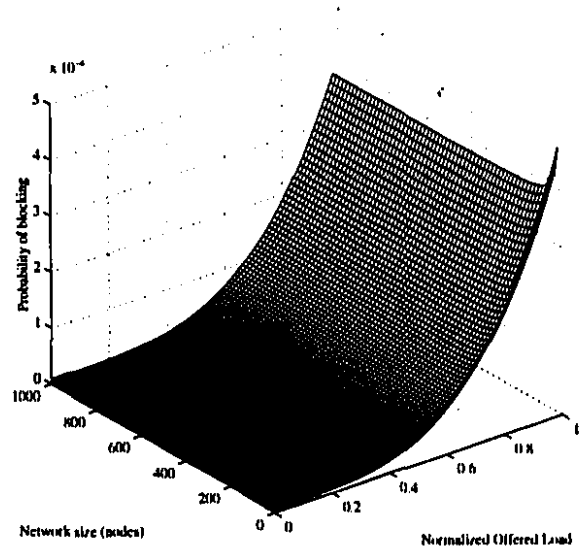


Figure 5.10: Three dimensional analysis space of the numerical analysis routine

If a set of graphs versus a wide range of network sizes (at full load) is desired, the number of increments of offered load should be set to a minimum, i.e., two. This minimizes the amount of unnecessary computation. Similarly, if a set of graphs versus offered load is desired (for a particular network size), the number of increments of offered load should be set high, e.g., 100 to obtain smooth curves. However, the range for the network sizes should be set to the smallest possible. For example, if curves versus offered load are needed for a network size of 64 nodes, the minimum network size should be set to 32, the maximum

network size to 64 and the number of increments for offered load to 100.[2] Even if both sets of curves, i.e., those versus network size and offered load are required it is often better to perform two separate analyses. Covering an entire two dimensional space of network size and offered load is unnecessary unless three dimensional plots are being generated.

5.5 System Requirements

The software tool was developed on a Unix platform using Matlab 4.2. The software encompasses over 150 routines totaling over 10000 lines of commented code. To illustrate a representative portion of the software, the first section of the appendix contains the Matlab code which calculates the performance measures for the dilated Crossout switch. The second section of the appendix contains the Matlab code for the calculation of blocking probability for the dilated Crossout switch in the linear hyperplane with an interleaved channel assignment scheme.

The memory requirements depend on the extent of the analysis which is user-defined. Calculations over 16 network sizes and 100 different offered loads result in 5 Mb data files. Regarding computation time, exact calculations for the linear hyperplane can be extremely intensive. As a representative figure, the calculations of the probabilities of blocking or acceptance for an $N=1024$ Crossbar, at a fixed offered load, take upwards of one minute to complete on Sun Sparc 5 workstation. The amount of time required for a complete analysis of the hyperplane depends on the architecture type, probability computation model and the range over which the design space exploration is carried out.

[2]The minimum and maximum network size cannot be set equal to each other due to software coding reasons.

Chapter 6

Numerical Results

6.1 Introduction

In order to assess the usefulness of the software tool, a design space exploration of the hyperplane is carried out. One of its intended applications is being the backbone of a large ATM switch with an aggregate bandwidth exceeding 1 Tbit/s. As such, the analysis is performed around an operating point of $P = 432$ bits per packet, $Z = 1024$ optical bit-channels in the single-stream, $B = 10^9$ clock cycle/s and a queue size of $Q_s = 32$ packets. The analysis examines the hyperplane supporting from $N = 16$ to $N = 1024$ nodes, over normalized offered loads from 0 to 1 and is presented in the following two sections.

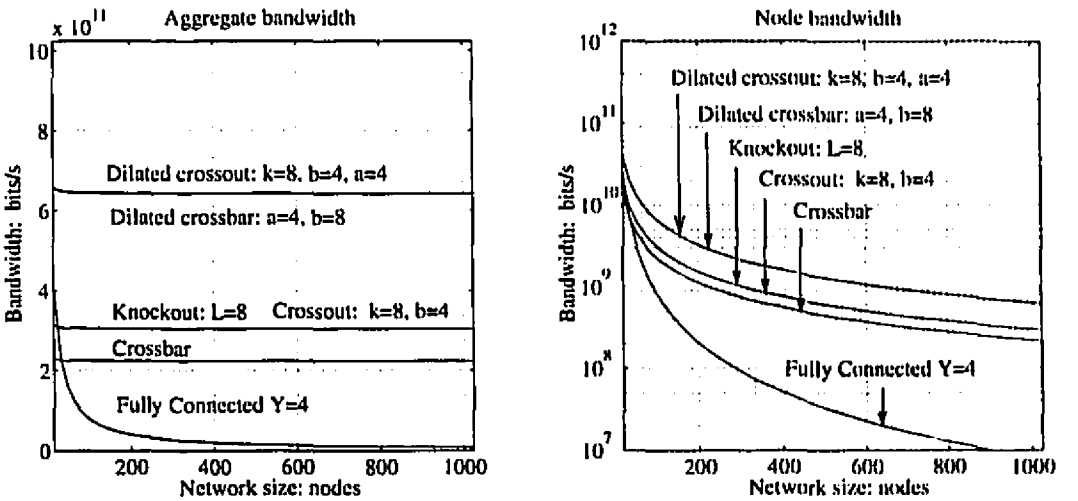
The results of the exploration for the linear and dual-stream hyperplanes are presented in the next two sections. The last section examines the impact of varying the hardware parameters of network size and the number of optical bit-channels and the operating points of offered load, clock frequency and packet size, upon the packet time-slot duration, aggregate bandwidth and their efficiencies.

As mentioned in Chapter 4, the analytic models in [25, 26, 27, 28] have been modified to reflect the final design of the June 1995 smart pixel array. The net effect the new design is that the propagation delay is "programmable" and the throughput can approach 1. The reader is referred to [25, 26, 27, 28] for the final analytic models. The reader is reminded, however, that the analyses presented in this thesis are concerned solely with the June 1994 smart pixel array design.

6.2 Linear Hyperplane

6.2.1 Unbuffered Analysis

The capacity of the linear hyperplane is a throughput of 1.024×10^{12} bits/s. The plot of aggregate bandwidth in Fig. 6.1a reveals that each of the six networks have throughputs that are below the maximum. The throughputs are virtually constant for all networks except the Fully Connected. The dilated Crossbar and dilated Crossout switches carry a greater amount of traffic than their single transmitter counterparts. The Fully Connected network has a throughput which is comparable to the other networks, but only for networks with a very small number of nodes. There are four factors which contribute to the unused capacity.



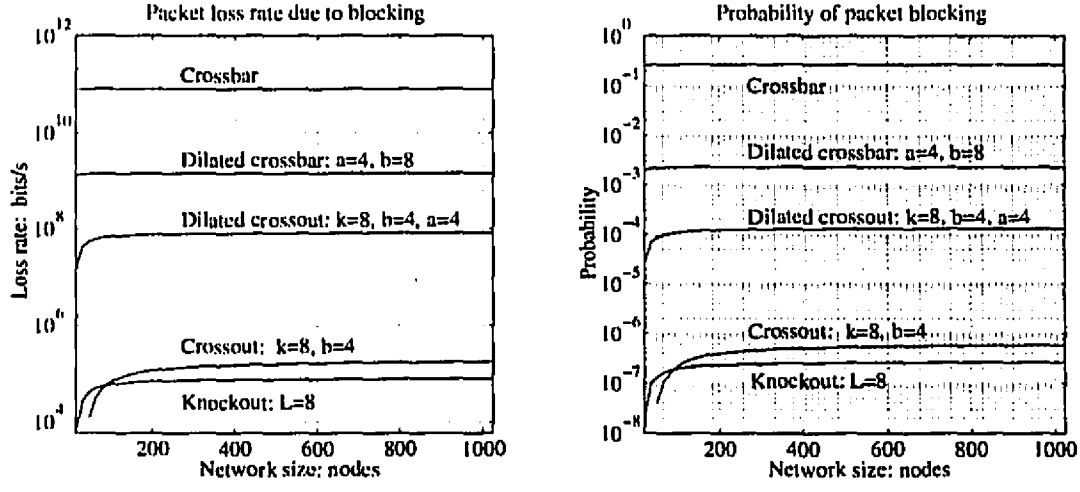
(a) aggregate bandwidth vs. network size

(b) packet time-slot duration vs. network size

Figure 6.1: Network throughput in the linear hyperplane using the June 1994 smart pixel array design

The first factor is the probability of blocking, shown in Fig. 6.2a, that arises from the inability of nodes to accept all the packets that may be received simultaneously. The dilated networks are seen to have greater blocking probabilities than those with a single transmitter per node. Since the former sources more packets there are greater numbers of packets arriving at any given destination. The Fully Connected network has a blocking

probability of identically zero and so never experiences any packet loss. The Crossbar has $PB = 0.26167$ which is less than $PB = e^{-1}(= 0.36788)$ as $N \to \infty$ for the classical Crossbar. This reduction in blocking is due to the embedding in the two separate streams of the linear hyperplane. For networks other than the Crossbar this is not the critical component for loss of throughput as the blocking tends to be relatively low. In other words, $PA > 0.99$; setting PA equal to one would only increase the throughput by less than 1%. The loss rates due to blocking shown in Fig. 6.2b, however, are directly proportional to PB. The Crossout and Knockout switches experience the least loss.



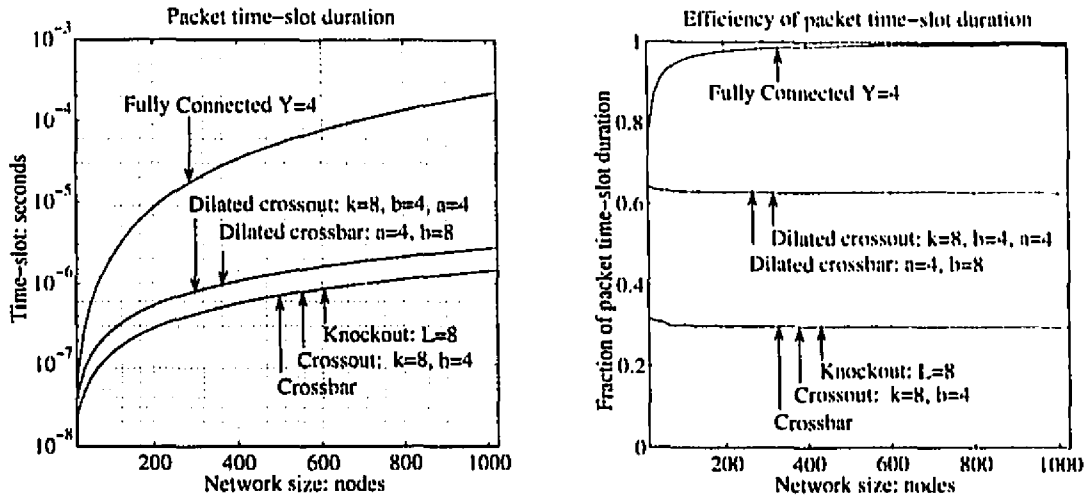
(a) loss rate vs. network size

(b) blocking probability vs. network size

Figure 6.2: Losses in the linear hyperplane using the June 1994 smart pixel array design

The second factor is the propagation delay portion of the packet time-slot duration for the networks shown in Fig. 6.3a. This constitutes an idle time after a packet has left its originating node is being passed through intermediate nodes while the destination node waits. The impact can be seen by considering Eqn. 6.2 for aggregate bandwidth at full load under the assumption that the propagation delay is zero. The result shows that the hypergraph networks except the Crossbar would have throughputs virtually equal to the capacity of the backplane.

$$\text{aggregate bandwidth} = \frac{aN(PA)P}{\frac{\lceil \frac{PaN}{b} \rceil}{B}} \text{ bits/s}$$



(a) packet time-slot duration vs. network size

(b) efficiency of time-slot vs. network size

Figure 6.3: Packet time-slot duration characteristics in the linear hyperplane using the June 1994 smart pixel array design

$$= (PA)ZB \text{ bits/s (Crossbar)} \quad (6.1)$$

$$\simeq ZB \text{ bits/s (other hypergraph networks)} \quad (6.2)$$

The unused capacity of the backplane results almost entirely from the propagation delay. The efficiency of the packet time-slot duration, i.e., the fraction which is transmission delay, is shown in Fig. 6.3b. A close correlation can be seen in comparing this graph with that of aggregate bandwidth for the dilated Crossbar, dilated Crossout, Knockout and Crossbar switches. The fraction of aggregate bandwidth over maximum capacity of the backplane is the same as the efficiency. The dilated networks are at approximately 60% efficiency, the Knockout at 30% efficiency and the Crossbar slightly lower due to its moderate blocking probability. While the Fully Connected network displays very high efficiency, its throughput is very low. The cause is addressed next.

The third factor concerns the usage of the edges embedded into the hyperplane. If a network has fewer transmitters than embedded edges, the number of packets that can be transmitted (in each packet time-slot duration) is limited by the number of transmitters available. Consequently, a number of edges will be idle. Depending on the connectivity of the particular network, this can cause the throughput to be several times smaller than it

otherwise would be. The only network which suffers this inefficiency is the Fully Connected network. With 4 transmitters per node, $\frac{N(N-1)}{2} - 4N$ edges are always idle which represents a severe under utilization of resources. The number of transmitters can be increased but not without additional hardware cost.

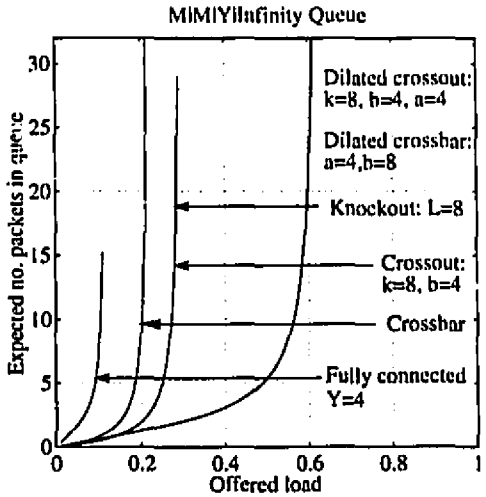
The fourth factor is the transmission delay which is given by $\lceil P/E \rceil$. If the packet size is not an integer multiple of the channel width, then an extra clock cycle is needed to transmit the last word of partially complete bits. This inefficiency is insignificant compared to the other three.

In order from the most to the least significant causes of low throughput are, propagation delay, the probability of blocking, the unused edges in the case of the Fully Connected network and the extra clock cycle needed to transmit a partially full channel.

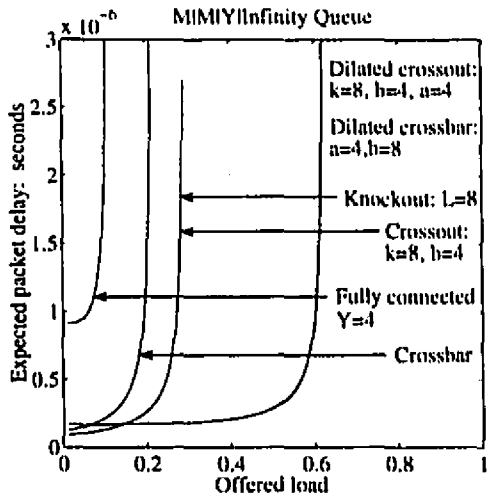
6.2.2 Buffered Analysis

The performance of the infinite size queue is seen in Fig. 6.4a,b. As the packet arrival rate approaches the rate at which the servers on the queue are able to handle the incoming packets the number of packets in the queue approaches infinity. For a given network the normalized offered load at which this occurs is analogous to the fraction of peak aggregate bandwidth which the network is able to carry. For example, the dilated Crossout has a bandwidth of approximately 0.65 Tbit/s which is 65% of the peak capacity of 1024 Gbit/s. It should be recalled that the analysis was performed using the June 1994 smart pixel array design. The final design of the June 1995 smart pixel array allows the throughput to approach 100%. The queue for the dilated Crossout is seen to saturate at a load of .6. Thus, the saturation point for each one of the networks reflects its throughput; the abscissa can be considered as the fraction of maximum throughput (1024 Gbits/s) rather than offered load. The expected packet delay through the queue is proportional to the packet time-slot duration. The delay is on the order of 1 μs when the offered load is below the saturation point.

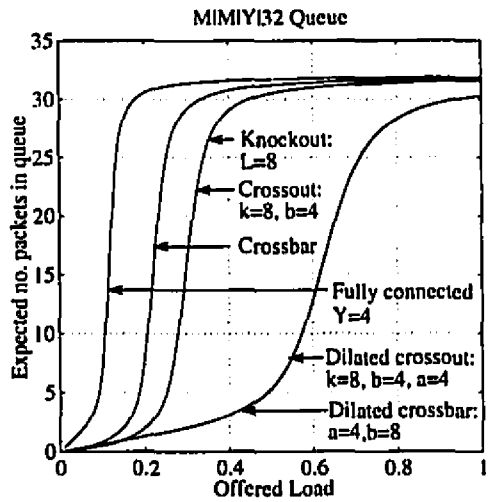
Fig. 6.4c,d plots the performance of the finite size queues which is similar to that of the infinite queue. When the offered load exceeds the saturation point, the queue reaches its maximum capacity of 32 packets. The expected packet delay peaks at approximately 32 times the packet time-slot duration. At this point, the packet loss occurs unlike the infinite



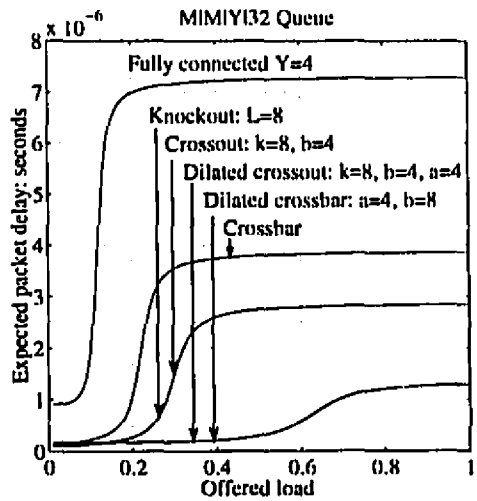
(a) Expected #packets in queue vs. offered load for infinite queue



(b) Expected packet delay vs. offered load for infinite queue

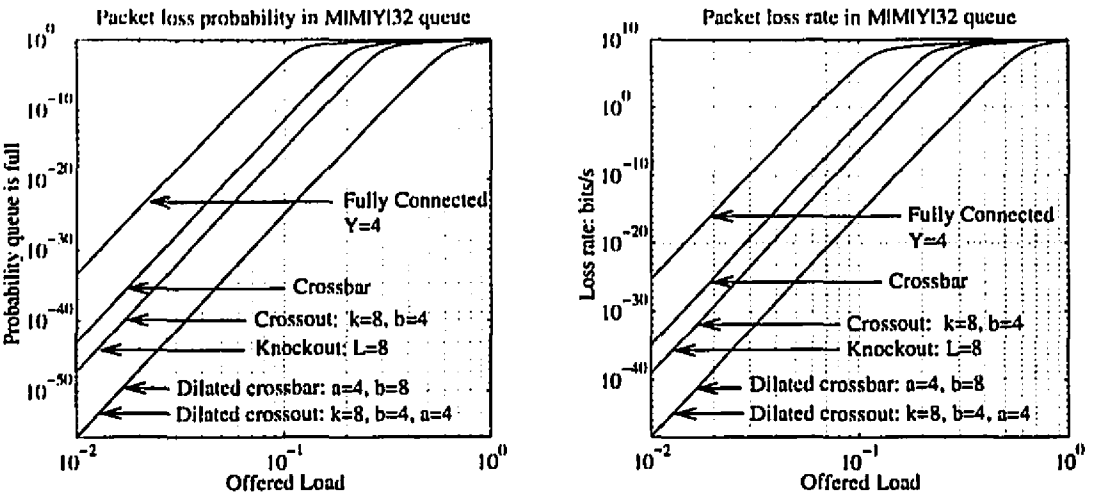


(c) Expected #packets in queue vs. offered load for finite queue



(d) Expected packet delay vs. offered load for finite queue

Figure 6.4: Buffered analysis of the linear hyperplane using the June 1994 smart pixel array design



(a) probability of queue overflow vs. offered load

(b) loss rate vs. offered load

Figure 6.5: Losses for buffered linear hyperplane using the June 1994 smart pixel array design

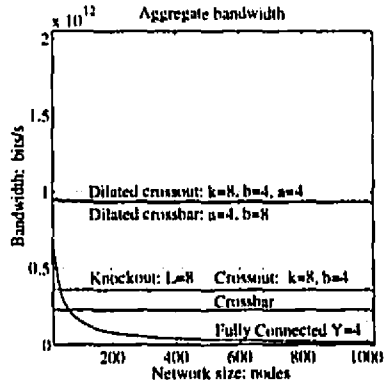
case which never loses packets. The probability of packet loss is shown in Fig. 6.5. The dilated networks are seen to have a very low loss probabilities and loss rates for offered loads that are less than 0.1. Even under full load the losses are low - this should be contrasted with the loss rates of Fig. 6.2a, for the unbuffered networks, which are several orders of magnitude larger.

6.3 Dual-Stream Circular Hyperplane

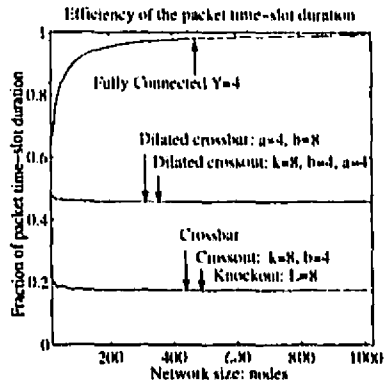
In order to compare the linear and dual-stream architectures, the latter is analyzed under the similar hardware configurations and same operating conditions. The number of optical bit-channels in a single-stream is held constant at $Z = 1024$. The packet size is held at $P = 432$ bits; the clock frequency is held at $B = 1$ GHz; the queue size is held at $Q_s = 32$ packets. The three embedding schemes described in Chapter 2 are analyzed below.

6.3.1 Maximized Edge Bandwidth Embedding

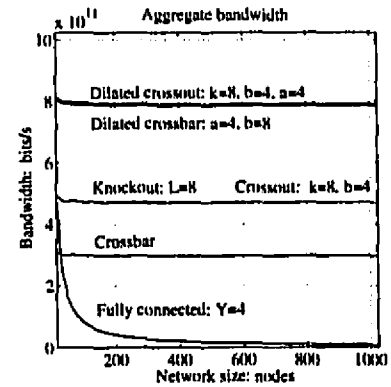
The maximized edge bandwidth embedding results in doubling the channel width of the embedded networks. The performance of the networks is shown in Fig. 6.6a,b. The dilated



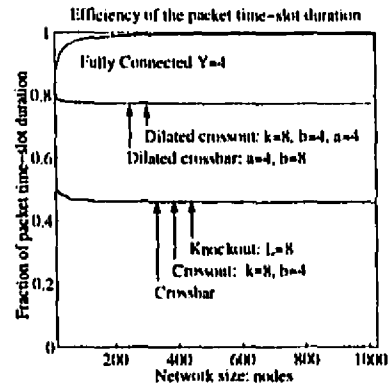
(a) Aggregate bandwidth vs. Network size
for the maximized edge bandwidth embedding



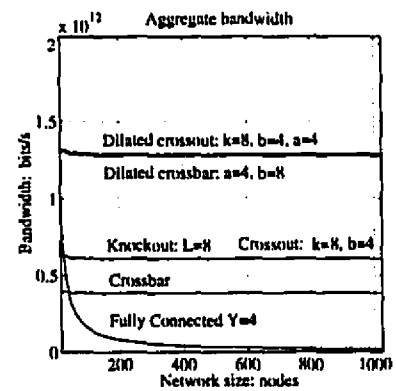
(b) Efficiency vs. Network size
for the maximized edge bandwidth embedding



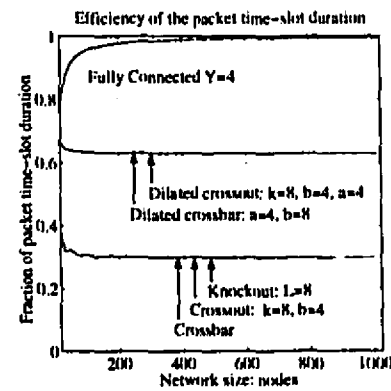
(c) Aggregate bandwidth vs. Network size
for the minimized propagation delay embedding



(d) Efficiency vs. Network size
for the minimized propagation delay embedding



(e) Aggregate bandwidth vs. Network size
for the optimized bandwidth and delay embedding



(f) Efficiency vs. Network size
for the optimized bandwidth and delay embedding

Figure 6.6: Unbuffered analysis of the dual-stream circular embeddings using the June 1994 smart pixel array design

networks experience the greatest gain in throughput (Fig. 6.6a) due to short packet time-slot durations. Their transmission delay is relatively large in comparison to the propagation delay and so derive the greatest benefit from the reduction of transmission delay. The Crossout and Knockout switches experience a similar gain, but smaller. This gain is due to their propagation delay being greater that the transmission delay for the linear case. The Crossbar, curiously, does not show any improvement at all. While the packet time-slot duration does decrease, there is an increase in the probability of blocking which offsets the potential gain in bandwidth. Recall that the Crossbar has a higher blocking probability in the (single-stream) circular hyperplane as opposed to the linear hyperplane (0.36 versus. 0.26). Overall, the throughputs are a lower fraction of the peak capacity, which is 2.048 Tbits/s, due to the lower transmission delays which decreases the efficiency of the time-slot duration (Fig. 6.6b). This decrease is seen in comparison to the graph of efficiency shown in Fig. 6.3b.

6.3.2 Minimized Propagation Delay Embedding

The minimized propagation delay embedding approximately halves the delay. The performance of the networks is shown in Fig. 6.6c,d. The throughputs (Fig. 6.6c) for all networks increase due to shorter packet time-slot duration. In contrast with the maximized edge bandwidth embedding, the dilated networks do not benefit as much as the Crossbar, Knockout and Crossout switches. The undilated networks have a relatively short transmission delay compared to their propagation delay and so derive the greatest benefit from the reduction in the latter. The dilated networks derive a similar benefit but less so. The throughputs are a greater fraction of the peak capacity, which is 1.024 Tbit/s, due to the higher efficiency of the time-slot (Fig. 6.6d) which is expected due to the shorter propagation delays. Compared with the maximized edge bandwidth embedding, the throughputs for the undilated networks are greater and that for the dilated networks is comparable.

6.3.3 Optimized Edge Bandwidth and Propagation Delay Embedding

The simultaneously maximized bandwidth and minimized propagation delay embedding results in doubling the edge bandwidth and approximately halving the delay. The performance of the networks is shown in Fig. 6.6e,f. The packet time-slot duration is the shortest

of all the embeddings. This results in approximately doubling the throughputs (Fig. 6.6e) when compared with the linear hyperplane embeddings. As a fraction of peak capacity which is 2.048 Tbit/s, the efficiency shown in Fig. 6.6f is almost identical to that of the linear case. The decrease in propagation delay which would raise the efficiency is offset by a comparable decrease in the transmission delay time. Despite this, from the point of view of usage of hardware resources this embedding scheme provides the highest throughput.

6.4 Effect of Hardware Parameters and Operating Conditions

As part of the design space analysis, five parameters are examined for their impact of the performance of the linear hyperplane. They are

- N: number of nodes on the backplane
- α: offered load
- Z: number of free-space optical bit-channels
- B: clock frequency of the backplane
- P: packet size in bits

The linear hyperplane architecture is assumed to support $N = 64$ nodes in the backplane, operate at a load of $\alpha = 1$, have $Z = 1024$ optical bit-channels in each stream, operate at a clock frequency of $B = 1.0$ GHz, and transmit packets of size $P = 432$ bits. In each of the following subsections, all the parameters are held constant except for one which is varied over a wide range. This allows the effects of one particular parameter to be isolated and studied carefully for its impact on throughput.

6.4.1 Effect of the Network Size on Performance

The aggregate bandwidth of the networks except for the Fully Connected network is practically independent of the number of nodes embedded in the hyperplane as shown in Fig. 6.1a. As N becomes large, i.e. $N \geq 64$ the aggregate bandwidth for the hypergraph networks is

given by Eqn. 6.3. For the dilated networks this value is approximately 0.643 Tbit/s and .304 Tbit/s for the Crossout and knockout networks.

$$\text{aggregate bandwidth} = \frac{\alpha a N(PA)P}{\frac{\left\lceil \frac{aPN}{Z} \right\rceil + (N-1)}{B}} \simeq \frac{\alpha a(PA)P \cdot B}{\frac{aP}{Z} + 1} \text{ bits/s/} \quad (6.3)$$

The Fully Connected network is different where increasing N severely degrades performance as its throughput is proportional to $1/N$ shown in Eqn. 6.4 and Fig. 6.1a. For a network size of $N = 16$ nodes, the throughput is 0.419 Tbit./s but drops to 7.97 Gbit/s for $N = 1024$ nodes.

$$\text{aggregate bandwidth} = \frac{\alpha N P}{\frac{\left\lceil \frac{N(N-1)P}{2Z} \right\rceil + (N-1)}{B}} \simeq \frac{\alpha P \cdot B}{\frac{NP}{2Z} + 1} \text{ bits/s} \quad (6.4)$$

Of equal importance is the node bandwidth (Fig. 6.1b). Since the aggregate bandwidth remains almost constant w.r.t. N for the hypergraph networks, the node bandwidth must then decrease as a function of $1/N$. The node bandwidth for the Fully Connected is similar but decreases as a function of $1/N^2$.

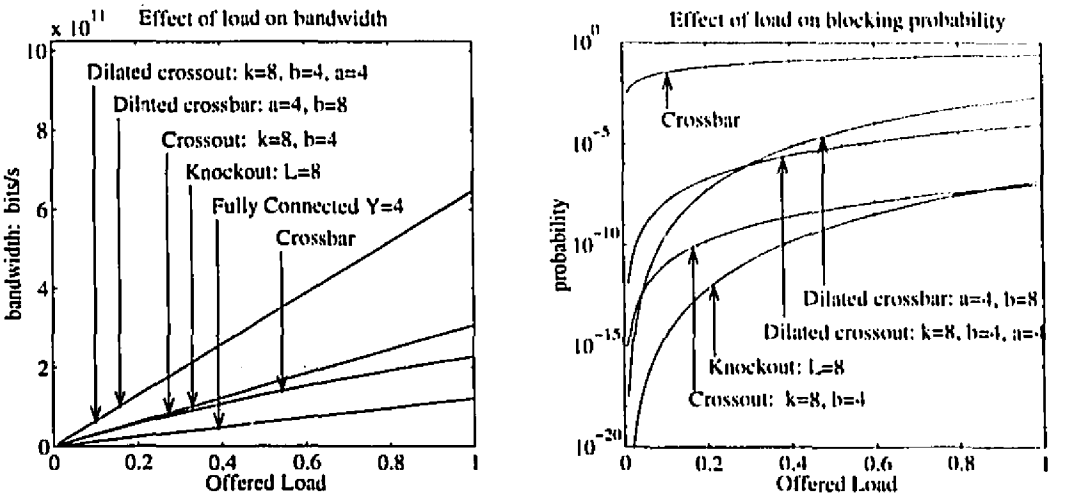
6.4.2 Effect of the Offered Load on Performance

A crucial aspect of a backplane's performance is the ability to carry traffic under different offered loads. The aggregate bandwidth for networks of size $N = 64$ is shown in Fig. 6.7a. The results show that the carried traffic is linearly dependent with the offered load. The blocking probability is shown in Fig. 6.7b. For offered loads below 40%, the blocking probability is below 10^{-5} for all networks except the Crossbar.

6.4.3 Effect of the Number of Optical Bit-Channels on Performance

Bandwidth is affected by the transmission delay portion of the packet time-slot duration. The transmission delay (in clock cycles) is given by is given by Eqn. 6.5, where $e = aN$ for hypergraph networks and $e = N(N-1)/2$ for the Fully Connected network.

$$\text{transmission delay} = \left\lceil \frac{Pe}{Z} \right\rceil \text{ clock cycles} \quad (6.5)$$



(a) aggegate bandwidth vs. offered load

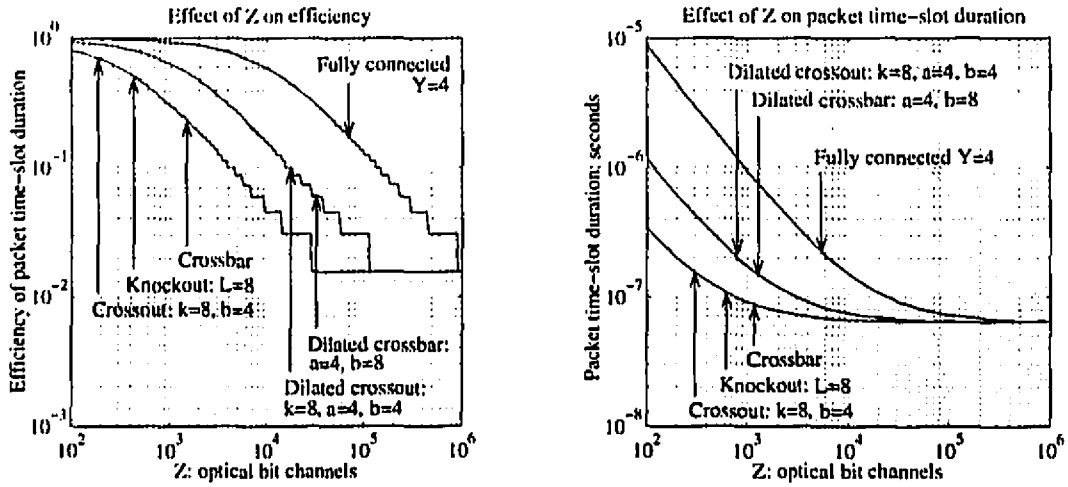
(b) blocking probability vs. offered load

Figure 6.7: Effect of offered load on bandwidth and blocking probability using the June 1994 smart pixel array design

If Z is increased to the point where it equals Pe, the transmission delay portion of the packet time-slot duration can be reduced to a single clock cycle. The channel width will have increased to the point where an entire packet can be transmitted in one clock cycle without having to be divided into several pieces. For hypergraph networks of 64 nodes with an input dilation of one, this occurs at $Z = 27648$ optical bit-channels. For the Fully Connected network, this occurs only at the much larger value of $Z = 870912$ optical bit-channels.

However, the propagation delay remains constant since, regardless of the channel width, a packet must still travel the length of the backplane, i.e., from node 1 to node N in the worst case. With a sufficiently large Z, the transmission delay reaches a lower bound of one. Increasing Z beyond this point is of no use as the packet time-slot duration consists almost entirely of propagation delay and is equal to N/B seconds, given by Eqn. 6.6, and is shown in Fig. 6.8b.

$$\begin{aligned} S &= \frac{\left\lceil \frac{Pe}{Z} \right\rceil + (N-1)}{B} \text{ seconds} \\ &= \frac{(1)+(N-1)}{B} \text{ seconds} \end{aligned}$$



(a) efficiency of packet time-slot duration vs. Z

(b) packet time-slot duration vs. Z

Figure 6.8: Effect of Z (optical bit-channels) on the packet time-slot duration using the June 1994 smart pixel array design

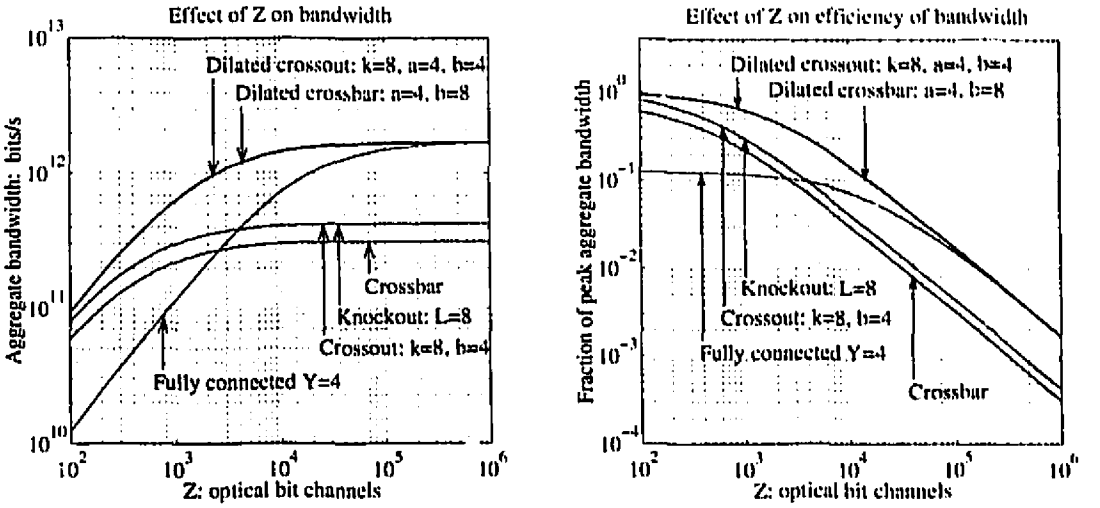
$$= \frac{N}{B} \text{ seconds} \tag{6.6}$$

The efficiency of the packet time-slot duration, given by Eqn. 6.7, reaches a lower limit of $1/N$ and is shown in Fig. 6.8a. It should be noted that this is an extreme case; such a configuration would not be used in practice.

$$\begin{aligned} \text{efficiency} &= \frac{\left\lceil \frac{P_c}{Z} \right\rceil}{\left\lceil \frac{P_c}{Z} \right\rceil + (N-1)} \\ &= \frac{1}{(1)+(N-1)} \\ &= \frac{1}{N} \end{aligned} \tag{6.7}$$

The transmission delay being equal to one, the aggregate bandwidth reaches a peak, given by Eqn. 6.8, and is shown in Fig. 6.9a. The aggregate bandwidth peaks at 0.173 Tbits/s for the dilated and fully connected networks, 0.432 Tbits/s for the Knockout and Crossout networks and 0.315 Tbits/ for the Crossbar.

$$\begin{aligned} \text{aggregate bandwidth} &= \frac{\alpha a N (PA) P}{\frac{1+(N-1)}{B}} \\ &= \alpha a (PA) P \cdot B \text{ bits/s} \end{aligned} \tag{6.8}$$



(a) aggregate bandwidth vs. Z (b) fractional bandwidth vs. Z

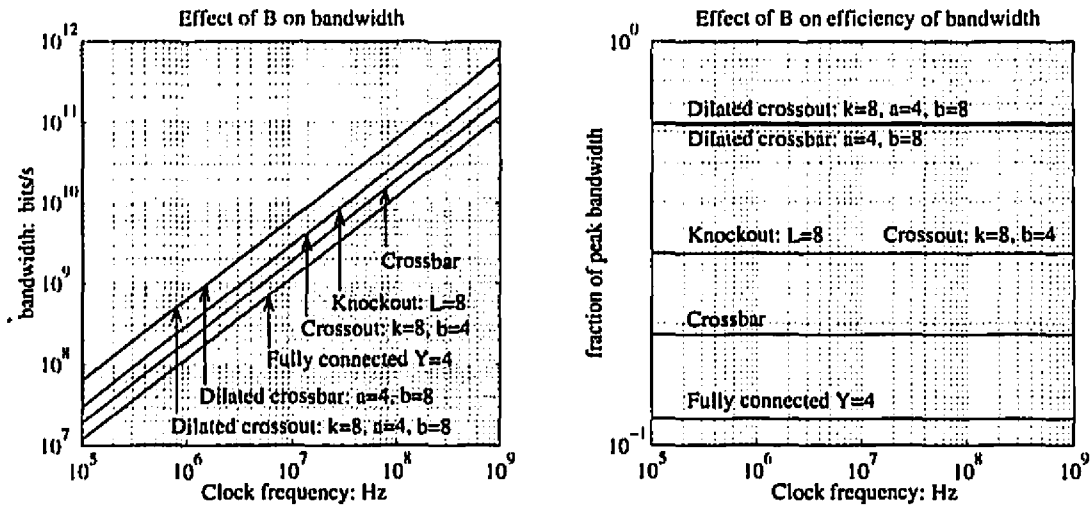
Figure 6.9: Effect of Z (optical bit-channels) on the aggregate bandwidth using the June 1994 smart pixel array design

The peak theoretical bandwidth of the backplane, ZB, increases linearly with the number of optical bit-channels. Increasing Z beyond Pe causes the fraction of peak capacity used to successfully deliver packets to decrease as $1/Z$. This fractional efficiency is given in Eqn. 6.9 and is shown in Fig. 6.9b.

$$\begin{aligned} \text{fractional efficiency} &= \frac{\text{aggregate bandwidth}}{ZB} \\ &= \frac{\alpha a N(PA)P}{ZB\left(\frac{\left\lceil \frac{Pe}{Z} \right\rceil + N - 1}{B}\right)} \\ &= \frac{\alpha a N(PA)P}{Z(1+(N-1))} \\ &= \frac{\alpha a(PA)P}{Z} \end{aligned} \tag{6.9}$$

6.4.4 Effect of the Clock Frequency on Performance

As is expected, any change in the clock frequency has a direct effect on the throughput. The relationship is one-to-one where any change in B results in exactly the same change in throughput. Fig. 6.10a shows that the aggregate bandwidth is directly proportional to the



(a) aggregate bandwidth vs. clock frequency

(b) fractional bandwidth vs. clock frequency

Figure 6.10: Effect of B (clock frequency) on the aggregate bandwidth using the June 1994 smart pixel array design

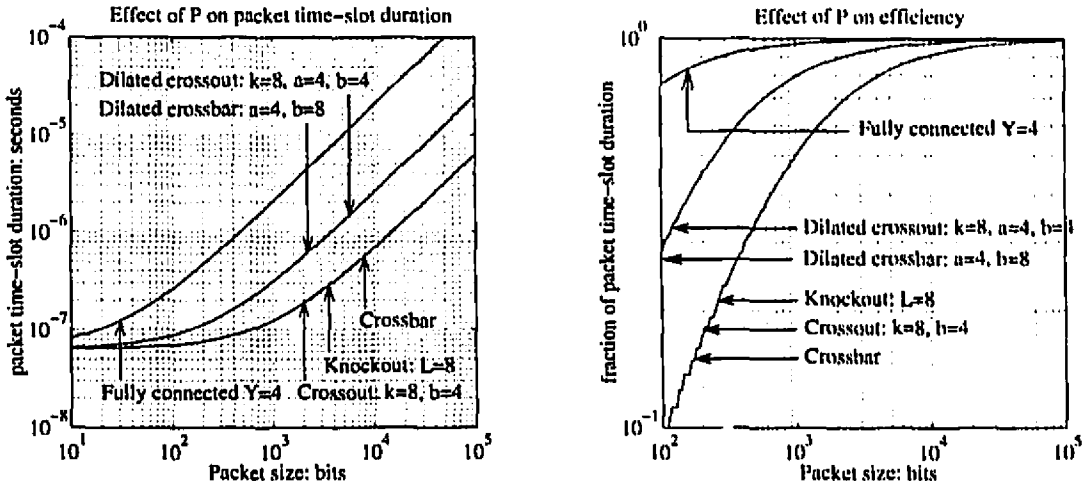
optical bit rate. In Fig. 6.10b, the carried traffic is seen to be a fixed fraction of the total capacity.

6.4.5 Effect of the Packet Size on Performance

As the packet size P is increased, Fig. 6.11a shows that the packet time-slot duration increases and does so almost linearly once the transmission delay is much larger than the propagation delay (Eqn. 6.10). The efficiency of the packet time-slot duration asymptotically approaches one (Eqn. 6.11) as shown in Fig. 6.11b.

$$\begin{aligned} S_{packet} &= \left\lceil \frac{Pe}{Z} \right\rceil + (N-1) \\ &\simeq \left\lceil \frac{Pe}{Z} \right\rceil \text{ clock cycles} \end{aligned} \tag{6.10}$$

$$\begin{aligned} \text{efficiency} &= \frac{\left\lceil \frac{Pe}{Z} \right\rceil}{\left\lceil \frac{Pe}{Z} \right\rceil + (N-1)} \\ &\simeq \frac{\left\lceil \frac{Pe}{Z} \right\rceil}{\left\lceil \frac{Pe}{Z} \right\rceil} \end{aligned}$$



(a) packet time-slot duration vs. packet size

(b) efficiency vs. packet size

Figure 6.11: Effect of P (packet size) on the packet time-slot duration using the June 1994 smart pixel array design

$$\simeq 1 \tag{6.11}$$

An increase in P is reflected in an increase in throughput due to the higher efficiency of packet transmissions. The aggregate bandwidth is given by Eqn. 6.12. The aggregate bandwidth for the dilated networks is almost equal to ZB bits/s which is the peak throughput of the hyperplane. At the current operating point of P=432 bits, the throughput is 0.650 Tbits/s for the dilated networks, 0.310 Tbits/s for the Crossout and Knockout switches and .235 Tbits/s for the Crossbar. From the graph of Fig. 6.12 it can be seen that any increase in packet size, from P = 432 bits, would produce a significant increase in throughput. By doubling the packet size to 864 bits, the throughput becomes 0.786 Tbit/s for the dilated networks, 0.468 Tbits/s for the Crossout and Knockout switches, and 0.350 for the Crossbar. Tripling the packet size to 1296 bits results in throughputs of .853 Tbits/s for the dilated networks, 0.574 for the Crossout and Knockout switches and 0.432 Tbits/s for the Crossbar. The Fully Connected network does not benefit greatly from an increase in packet size. The efficiency of its packet time-slot duration is quite high due its very long transmission delay and is not affected much by increasing packet size. As mentioned previously, its low throughput is due to the large number of unused edges.

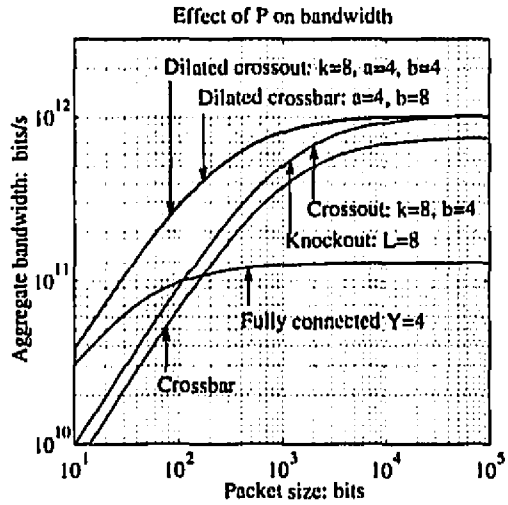


Figure 6.12: Effect of P (packet size) on the aggregate bandwidth using the June 1994 smart pixel array design

$$\begin{aligned} \text{aggregate bandwidth} &\simeq \frac{\alpha a N(PA)P}{\frac{aNP}{\frac{Z}{B}}} \\ &\simeq \alpha(PA)ZB \\ &\simeq (PA)ZB \text{ bits/s} \quad \text{at full load} \end{aligned} \tag{6.12}$$

6.5 Summary

The numerical results of a design space exploration of the linear and dual-stream circular hyperplanes were presented in this chapter. The analysis was performed around an operating point of a packet size of $P = 432$ bits, $Z = 1024$ optical bit-channels in the single-stream, a clock frequency of $B = 1.0$ GHz and a queue size of $Q_s = 32$ packets. The analysis examined the hyperplanes supporting from $N = 16$ to $N = 1024$ nodes, over normalized offered loads from 0 to 1.

The impact of varying the hardware parameters of network size and the number of optical bit-channels and the operating points of offered load, clock frequency and packet size, upon the packet time-slot duration, aggregate bandwidth and their efficiencies was

then examined. Each parameter was varied one at a time, while holding the remaining ones constant. In this manner, the effects of each parameter on throughput were isolated.

As a result of the design space exploration, an efficient architecture and operating point for the hyperplane were identified.

Chapter 7

Conclusions

The analytic model presented in [25, 26] describing the hyperplane architecture [27, 28] has been implemented in software. This software has provided a tool which greatly facilitates design space explorations of the hyperplane by automating the numerical analysis and subsequent visualization of the results.

Validation of the software tool is two-fold. Firstly, the numerical analysis portion of the tool allows the user to specify the:

- hyperplane architecture: linear or dual-stream circular
- network embedding scheme for the dual-stream circular hyperplane
  - maximized edge bandwidth embedding
  - minimized propagation delay embedding
  - optimized edge bandwidth and propagation delay embedding
- channel assignment scheme for the linear hyperplane
  - sequential assignment
  - interleaved assignment
- probability computation model
  - exact binomial model
  - approximate binomial model
  - Poisson model
- operating point: clock frequency, packet size, offered load, queue size, etc.
- the range and granularity of the analysis

Secondly, the numerical visualization portion of the tool allows the user to visualize the results in a manner such that a deeper insight can be gained into the operation of the hyperplane. These results have helped in identifying a more efficient operating point.

The software was used to perform a design space exploration of the hyperplane to evaluate its suitability for applications such as large scale ATM switches. The analysis covered the unbuffered and buffered linear hyperplanes, the maximized edge bandwidth, the minimized propagation delay, the simultaneously optimized edge bandwidth and propagation delay embeddings in the dual-stream circular hyperplane. From the results of the analysis it can be concluded that:

1. The dual-stream circular hyperplane with the optimized edge bandwidth and propagation delay embedding provides the highest throughput. The throughput is approximately twice that of an equivalent linear hyperplane. However, the latter does not require the extra bulk optics need to form a ring [26, 28].

2. The addition of buffering with FIFO input queues greatly reduces the packet loss rate.

3. The throughput of the embedded networks in the hyperplane can be improved by modifying the packet size to the number of optical bit-channels ratio. This ratio should be made as large as possible relative to the number of nodes in the network. This increase has the effect of maximizing the transmission delay and minimizing the propagation delay, which makes packet transmissions between nodes more efficient in their use of the logical channels.

4. The novel interleaved channel assignment scheme for the Crossout and dilated Crossout switches produces a significant reduction in the probability of packet blocking and hence loss rate. Depending on the particular configuration of the switch, i.e., the number of slices, channels per slice, transmitters per node and receivers per slice, the improvement may be an order of magnitude. However, the increase in throughput is negligible.

In conclusion, it can be said the software tool is invaluable for performing design space explorations which will aid the continued evolution of the free-space photonic backplane.

Bibliography

[1] "Canadian Institute for Telecommunications Research, Annual Report," 1994. Available from McGill University by calling (514) 398-8104.

[2] R. Nordin, A. Levi, R. Nottenburg *et al.*, "A systems perspective on digital interconnection technology", *IEEE Journal of Lightwave Technology*, vol. 10, no. 6, 1992.

[3] D. Bertsekas, R. Gallager, "Data Networks, 2nd Edition", Prentice Hall, Englewood Cliffs, New Jersey, 1992.

[4] P. G. Harrison, N. M. Patel, "Performance Modelling of Communication Networks and Computer Architecture", Addision Wesley, 1993.

[5] J. E. Midwinter, "Photonic switching technology: component characteristics versus network requirements", *IEEE Journal of Lightwave Technology,* vol. 6, no. 10, pp. 1512-1519, 1988.

[6] M. R. Feldman, S. C. Esener *et. al.*, "Comparison between optical and electrical interconnects based on power and speed considerations", *Applied Optics,* vol. 27, no. 9, pp. 1742-1751, 1988.

[7] A. Guha, J. Bristow, C. Sullivan, A. Hussain, "Optical interconnections for massively parallel architectures", *Applied Optics,* vol. 29, no. 8, pp. 1077-1093, 1990.

[8] P. Haugen, S. Rychnovsky , A. Hussain, L. Hutcheson, "Optical interconnects for high speed computing," *Optical Engineering,* vol. 25, no. 10, pp. 1076-1085, 1986.

[9] D. H. Hartman, "Digital high speed interconnects: a study of the optical alternative," *Optical Engineering,* vol. 25, no. 10, pp. 1086-1102, 1986.

[10] AT&T, FET-SEED Technical Specifications, AT&T FET-SEED Workshop, August 1993, Newark, New Jersey.

[11] *IEEE standard 896.2-1991: Futurebus+ Physical Layer and Profile Specification.*

[12] A. Dickinson, M. E. Prise, "An Integrated Free-Space Optical Bus," *IEEE International Conference on Computer Design: VLSI in Computers and Processors*, pp. 62-65, 1989.

[13] F. B. McCormick, T. J. Cloonan, A. L. Lentine *et al.*, "Five-stage free-space optical switching network with field-effect transistor self-electro-optic-effect-device smart-pixel-arrays," *Applied Optics*, vol. 33., no. 8, pp. 1601-1618, 1994.

[14] R. T. Chen, H. Lu, it et al., "60 GHz board to board optical interconnection using polymer optical buses in conjunction with microprism couplers," *Applied Physics Letters*, vol. 60, iss. 5, pp. 536-538, 1992.

[15] K. Hamanaka, "Optical bus interconnection system using Selfoc lenses,", *Optics Letters*, vol. 16, no. 16, pp. 1222-1224, 1991.

[16] K. Hamanaka, "Integration of free-space interconnects using selfoc lenses: image transmission properties," *Japanese Journal of Applied Physics*, vol. 31, no. 5B, pp. 1656-1662, 1992.

[17] M. Feldman, "Holographic optical interconnects for multichip modules," *Electronic Engineering*, vol. 64, iss. 768, pp. 49-50, 53, 1992.

[18] R. C. Kim, E. Chen, F. Lin, "An Optical Holographic Backplane Interconnect System," *Journal of Lightwave Technology*, vol. 9, no. 12, pp. 1650-1656, 1991.

[19] Y. Yeh, M. G. Hluchyj, A. S. Acampora, "The Knockout switch: a simple, modular architecture for high-performance packet switching," *IEEE J. Select. Areas Commun.*, vol. 5, no. 8, pp. 1274–1283, 1987.

[20] H. S. Hinton, A. L. Lentine, "Multiple Quantum-Well Technology", *IEEE Circuits and Devices*, pp. 12-18, March 1993.

[21] T. H. Szymanski, H. S. Hinton, "Graph Embeddings in a Photonic HyperPlane", *International Conference on Applications of Photonic Technology (ICAPT-94)*, June, 1994.

[22] T. H. Szymanksi, H. S. Hinton, "Architecture of a terabit free-space photonic backplane," *International Conference on Optical Computing - 94*, August, 1994.

[23] T. H. Szymanksi, "Intelligent optical backplanes," *International Conference on Optical Computing - 95*, March, 1995.

[24] T. H. Szymanski, H. S. Hinton, "Smart pixel designs for a terabit free-space photonic backplane," *IEEE/LEOS Summer Topical Meeting '94*, July, 1994.

[25] T. H. Szymanski, H. S. Hinton, "Embedding nonblocking networks in a photonic hyperplane", Submitted.

[26] T. H. Szymanski, H. S. Hinton, "Embedding nonblocking networks in a circular photonic hyperplane", Submitted.

[27] T. H. Szymanski, H. S. Hinton, "Architecture of a terabit free-space photonic hyperplane", Conditionally accepted for publication, JPCD.

[28] T. H. Szymanski, H. S. Hinton, "Architecture of a free-space circular photonic hyperplane", Submitted.

[29] J. Walrand, *An Introduction into Queuing Networks*, Prentice Hall International Editions, 1988.

[30] T. G. Robertazzi, *Computer Networks and Systems*, Springer-Verlag, 1994.

[31] K. Kant, *Introduction to Computer System Performance Evaluation*, McGraw-Hill, 1992.

[32] L. Kleinrock, *Queueing Systems Volume 1: Theory*, John Wiley & Sons, 1975.

[33] T. H. Szymanski, "Hyper-meshes: optical interconnection networks for parallel computing," *Journal of Parallel and Distributed Computing*, To appear April, 1995.

Appendix A

Numerical Analysis Software

A.1 Top Level Routine for the Dilated Crossout Switch

The top level routine which analyzes the dilated Crossout switch is presented below. The routine calls a number of subroutines which calculate various quantities such as the packet time-slot duration and the probability of blocking.

```
% ------------------------------------------------------------- %
%                 Dilated Crossout
% ------------------------------------------------------------- %

fprintf('\nDilated crossout\n')

a_dxout = 4;    % four transmitters per node
b_dxout = 4;    % output dilation
k_dxout = 8;    % 8 slices
Y_dxout = 4;

for x = 1:N_points

fprintf('Network size = %d nodes (%d of %d)\n',x*N_step,x,N_points);

N = x*N_step;       % let number of nodes range from [N_low:N_high]
C = a_dxout*N/k_dxout;    % let a_dxout*N = C*k
```

```
% ------------------------------------ %
% derivation of packet time slot duration
%   - transmission delay, T
%   - propagation delay, P
%   - packet time-slot duration, S
%   - inefficiency of packet time-slot
%   - efficiency of packet time-slot
% ------------------------------------ %

edges_dxout    = a_dxout*N;

T_dxout(x)     = transmission_delay(B,P,Z,edges_dxout);
P_dxout(x)     = propagation_delay(HP,scheme,N,B);

[S_dxout(x), ...
 ineff_dxout(x), ...
 eff_dxout(x) ] = packet_time_slot(T_dxout(x),P_dxout(x));

% ------------------ %
% peak edge bandwidth
% ------------------ %

E_dxout(x) = edge_bandwidth(Z,B,P,edges_dxout);

% ---------------------------------------------------------- %
% carrying capacity of dilated crossout embedded in hyperplane:
% the throughput at full load in the absence of blocking
% ---------------------------------------------------------- %

capacity_dxout(x) = capacity(a_dxout,N,P,S_dxout(x));

% -------------------- %
% expected service time
% -------------------- %

PA = acceptance(HP,accuracy,channel,N,C,k_dxout,a_dxout,b_dxout,1);
mu_dxout(x) = a_dxout*PA/(Y_dxout*S_dxout(x));

for y = 1:alpha_max,
alpha = y/alpha_max;

    % ------------------------------------------- %
    % probabilities of packet acceptance and blocking
    % ------------------------------------------- %
```

```
PB_dxout(x,y) = blocking(HP,accuracy,channel,...
     N,C,k_dxout,a_dxout,b_dxout,alpha);

PB = PB_dxout(x,y);

PA_dxout(x,y) = 1 - PB_dxout(x,y);
PA = PA_dxout(x,y);

% ------------------------- %
% aggregate, node bandwidth
% ------------------------- %

[aggBW_dxout(x,y),nodeBW_dxout(x,y)] = ...
     bandwidth(a_dxout,alpha,N,PA_dxout(x,y),P,S_dxout(x));

% -------------------------- %
% unused capacity, loss rate
% -------------------------- %

[unused_dxout(x,y),loss_dxout(x,y)] = ...
     loss_unbuffered(alpha,Z,B,aggBW_dxout(x,y),PB,PA);

% ------------------------------------------- %
% lambda = offered load in packets/(node*seconds)
% rho = normalized load
% ------------------------------------------- %

lambda         = alpha*lambda_max(x);
rho_dxout(x,y) = lambda/(Y_dxout*mu_dxout(x));

% ------------------------------------------------------ %
% expected number of packets in the infinite queue
% expected number of packets in the finite queue (size Qs)
% expected delay of a packet in the infinite queue
% expected delay of a packet in the finite queue
% rate of packet loss in finite queue
% probability of packet loss in finite queue
% throughput of infinite queue
% throughput of finite queue
% ------------------------------------------------------ %

[Exp_C_inf_dxout(x,y), ...
 Exp_C_Qs_dxout(x,y), ...
```

```
        Exp_D_inf_dxout(x,y), ...
        Exp_D_Qs_dxout(x,y), ...
        lr_Qs_dxout(x,y), ...
        lp_Qs_dxout(x,y), ...
        tp_inf_dxout(x,y), ...
        tp_Qs_dxout(x,y)] = qstate(Qs,lambda,mu_dxout(x),Y_dxout);

    end    % for y = 1:alpha_max

end    % for x = 1:N_points
```

A.2 Probability of Blocking Routine for the Dilated Crossout Switch

The routine which calculates the probability of blocking for the dilated Crossout switch in the linear hyperplane under the interleaved channel assignment scheme is presented below. This routine is an implementation of Eqn. 4.32. The routine takes advantage to loop unrolling techniques.

```
%-------------------------------------------------------------------%
%
%  Unbuffered Dilated Crossout Switch
%    Binomial Formulation
%
%-------------------------------------------------------------------%

% N     = number of nodes in the Hyperplane
% C     = each optical crossbar slice has C channels
% k     = number of crossbar slices on each smart pixel array
% a     = the number of transmitters per node; also input dilation
% b     = each slice can extract b packets simultaneously
% alpha = Prob[node has a packet to transmit]

function x = pb2i_other_bin_fast(N,C,k,a,b,alpha)

% The acronym for naming the function stands for:
%
% pb   : probability of blocking ('pa' would indicate probability
%        of acceptance)
```

```
% 2    : for the two streams of the linear hyperplane
%        (no number is used for the single-stream circular hyperplane)
% i    : interleaved channel assignment ('s' would indicate the
%        sequential channel assignment scheme)
% other: assumes that a node does not transmit to itself, i.e.,
%        transmits only to other nodes
% bin  : binomial model ('pois' would indicate the Poisson model)
% fast : the routine optimized for speed by loop unrolling
%        ('apprx' would indicate the approximate model
%         'slow' would indicate the slow and direct implementation)


% Returns the conditional probability of acceptance for the
% dilated Crossout switch in the Hyperplane, based on the binomial model

if C*k ~= a*N,
   error(' C*k does not equal aN'),
end

% ------------------------------------------------------ %
% if the output dilation, b, per slice is  greater than
% the number of channels on that slice, then all
% packets will always be received; having b > C is
% pointless since only up to C packets will ever arrive
% ------------------------------------------------------ %

if C<b,
   fprintf(' Warning: C should be greater and or equal to b\n');
   x=0;
   return
end

% ~~ all packets always accepted ~~ %

if C==b,
   x=0;
   return
end


% ------------------------------------------------------------ %
% in all other cases, must compute the acceptance probability
% ------------------------------------------------------------ %
```

```
% ------------------------------------ %
% this optimized version does not return
% valid results if a>1 -- call
% the approximate version instead
% ------------------------------------ %
if a>1,
   x=pb2i_other_bin_aprx(N,C,k,a,b,alpha);
   return
end

% ---------------------------------------------- %
% let's handle the cases where the slices are full
% ---------------------------------------------- %

here = alpha./(N-1); % store repeatedly used value
not_here = 1 - here; % store repeatedly used value
balance = here/not_here; % store repeatedly used value

% ------------------------------------------------------------ %
% compute and store the binomial values that will be needed;
% each successive combination can be computed from the previous
% value by multiplying by (C-j+1) and dividing by j
% each successive (alpha/N)^j * (1-alpha/N)^(C-j) term can be
% computed by starting with (1-alpha/N)^C and then multiplying
% for each j by (alpha/N) / (1-alpha/N)
% ------------------------------------------------------------ %

partial_W = zeros(1,C);

for W = b+1:C

W1 = W+1;
binomial = zeros(1,W1); % dimension binomial vector
binomial(1) = 1*not_here^W; % initial value of binomial at j=0

% ------------------------------------------ %
% precompute value of binomial at j=b; this is
% used as a reference value to break out of the
% loop below which calculates all the binomials
% ------------------------------------------ %

binomial(b+1) = choose(W,b)*here^b*not_here^(W-b);
```

```
% ------------------------------------------------------------ %
% compute and store the binomial values that will be needed;
% each successive combination can be computed from the previous
% value by multiplying by (W-j+1) and dividing by j
% ------------------------------------------------------------ %

for j = 1:W1,

   binomial(j+1) = binomial(j)*(W1-j)/j * balance;

   % ------------------------------------------------ %
   % if the current binomial is 15 orders of magnitude
   % less than the largest relevent binomial, i.e.
   % binomial(b+1) then exit the loop;
   %
   % Using this ratio to exit the loop is better than
   % setting a fixed limit, since it will accomodate
   % any pb to be calculated no matter how low
   % ------------------------------------------------ %

   if binomial(j+1)/binomial(b+1) < 1e-15 break; end

end

% ---------------------------------- %
% for b+1 to W packets arriving only
% b packets can be received
% ---------------------------------- %

for j = b+1:W,
   partial_W(W) = partial_W(W) +(j-b)*binomial(j+1);
end

end % for W = 1:C1

partial_slices = k*k;
full_slices = k*(k+1)/2 - k;

packets_partial_slice = partial_slices * sum(partial_W(b+1:C-1)) + ...
full_slices*partial_W(C);

x = 2/(a*alpha*N) * packets_partial_slice;
```