INFORMATION TO USERS

This was produced from a copy of a document sent to us for microfilming. While the most advanced technological means to photograph and reproduce this document have been used, the quality is heavily dependent upon the quality of the material submitted.

The following explanation of techniques is provided to help you understand markings or notations which may appear on this reproduction.

1. The sign or "target" for pages apparently lacking from the document photographed is "Missing Page(s)". If it was possible to obtain the missing page(s) or section, they are spliced into the film along with adjacent pages. This may have necessitated cutting through an image and duplicating adjacent pages to assure you of complete continuity.

2. When an image on the film is obliterated with a round black mark it is an indication that the film inspector noticed either blurred copy because of movement during exposure, or duplicate copy. Unless we meant to delete copyrighted materials that should not have been filmed, you will find a good image of the page in the adjacent frame.

3. When a map, drawing or chart, etc., is part of the material being photographed the photographer has followed a definite method in "sectioning" the material. It is customary to begin filming at the upper left hand corner of a large sheet and to continue from left to right in equal sections with small overlaps. If necessary, sectioning is continued again—beginning below the first row and continuing on until complete.

4. For any illustrations that cannot be reproduced satisfactorily by xerography, photographic prints can be purchased at additional cost and tipped into your xerographic copy. Requests can be made to our Dissertations Customer Services Department.

5. Some pages in any document may have indistinct print. In all cases we have filmed the best available copy.

University
Microfilms
International
300 N. ZEEB ROAD, ANN ARBOR, MI 48106
18 BEDFORD ROW, LONDON WC1R 4EJ, ENGLAND
Dias, Daniel Manuel

Packet Communication in Delta and Related Networks

Rice University

University Microfilms International

300 N. Zeeb Road, Ann Arbor, MI 48106

Ph.D. 1981
RICE UNIVERSITY

Packet Communication in Delta and Related Networks

by

Daniel M. Dias

A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy

APPROVED, THESIS COMMITTEE:

J. R. Jump,
Professor of Computer Science in
the Department of Electrical Engineering
Chairman

J. B. Sinclair,
Assistant Professor of Computer Science in
the Department of Electrical Engineering

P. E. Pfeiffer,
Professor of Mathematical Science

Houston, Texas
April, 1981
ABSTRACT

PACKET COMMUNICATION IN DELTA AND RELATED NETWORKS

by

Daniel M. Dias

Delta networks are a class of multistage interconnection networks that can be used to interconnect a large number of modules in a modular computing system. In a packet communication environment, the modules in a system communicate asynchronously using fixed size packets. In this environment, delta networks exhibit a large parallelism in packet transfer leading to good performance. They also allow for local control in packet routing, extensibility, clean interfaces, speed independence, and simple proofs of correctness. They can be augmented to improve reliability, and pruned to have different numbers of input and output links. Feedback paths can be added to provide a range of performance.

Special extreme cases of \((N \times N)\) delta networks are the full crossbar switch, which has \(O(N^2)\) gates and \(O(N)\) ideal bandwidth, and the single bus, which has \(O(N)\) gates and \(O(1)\) bandwidth. Delta networks constructed from fixed size basic switches have \(O(N \log N)\) gates and \(O(N)\) ideal bandwidth. The performance of delta networks with fixed size basic switches is comparable to that of the full crossbar switch in realistic environments as well. The class of delta networks includes several other networks that have been proposed in the literature. Thus, this study applies to these networks as well.

Unbuffered delta networks have no internal buffers within the network. A simple model for network operation can be used to obtain a
network characteristic that can then be used to obtain good estimates of performance for other models and environments. Using priority schemes in arbitrating between conflicting packets (that require passage through the same link at the same time) leads to a deterioration in performance as compared to arbitration with equiprobable selection from conflicting packets.

Inserting a buffer between the stages of a delta network leads to a considerable improvement in performance, making its performance better than that of the (same size) unbuffered crossbar switch. However, as the number of buffers between stages is increased, the network bandwidth saturates quickly to a constant, while the delay that a packet encounters in the network increases almost linearly with buffer size. Thus, the buffer size for most practical applications should be limited to one or two. The performance obtained is also sensitive to precisely how the switches in the network operate. The switch operation policy needs to be chosen depending on the values of certain network parameters in an implementation. There are many ways of pruning delta networks. Optimal pruned delta networks, with buffers between stages, have a nonintuitive topology.

Introducing feedback paths leads to a range of performance and cost. However, packets can deadlock in such networks. It is possible to detect and recover from deadlocks. For good performance to be obtained, it is necessary to prevent the frequent occurrence of deadlocks. This can be achieved by controlling the flow of packets into the networks.
ACKNOWLEDGEMENTS

I sincerely thank Dr. J. R. Jump, my thesis adviser, for his guidance, interest, encouragement, and constructive criticism. I am grateful to Dr. J. B. Sinclair, and Dr. P. E. Pfeiffer, for serving on my thesis committee.

I thank my friends, T. P. Ananthnarayan, John Dooley, Abhijit Gadgil, Joseph Gonsalves, Tim Gonsalves, Shirish Gunaji, Bala Iyer, Anand Jagannathan, Charlie Loeffler, Manoj Kumar, G. A. Merchant, K. V. Bapa Rao, and Bart Sinclair, among many others, for making my graduate study at Rice enjoyable.

Most importantly, I thank my wife, Lakshmi, for her support, encouragement, and endurance, particularly during the last lap that seemed interminable. This thesis is dedicated to her.

This research has been supported by the National Science Foundation under grants DCR-7414283 and MCS-8001667.
TABLE OF CONTENTS

CHAPTER 1  INTRODUCTION AND BACKGROUND  1
1.1 Introduction  1
1.2 Background and Related Work  6
1.3 Plan of the Thesis  18

CHAPTER 2  DELTA AND RELATED NETWORKS: TOPOLOGY AND PROPERTIES  20
2.1 Introduction  20
2.2 Delta Networks and their Properties  22
2.3 Bidirectional Delta Networks  42
2.4 Augmented Delta Networks  49
2.5 Pruned Delta Networks  54
2.6 Chapter Summary  60

CHAPTER 3  UNBUFFERED DELTA NETWORKS  63
3.1 Introduction  63
3.2 The Model and Performance Measures  64
3.3 \((b^n \times b^n)\) Delta Networks  68
3.4 Pruned and Augmented Delta Networks  90
3.5 Chapter Summary  98

CHAPTER 4  BUFFERED DELTA NETWORKS  101
4.1 Introduction  101
4.2 The Environment and Performance Criteria  103
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.3</td>
<td>( (2^n \times 2^n) ) Buffered Delta Networks</td>
<td>105</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Model (i)</td>
<td>106</td>
</tr>
<tr>
<td>4.3.1.1</td>
<td>Analysis</td>
<td>108</td>
</tr>
<tr>
<td>4.3.1.2</td>
<td>Simulation</td>
<td>156</td>
</tr>
<tr>
<td>4.3.1.3</td>
<td>Discussion and Comparison</td>
<td>158</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Model (ii)</td>
<td>171</td>
</tr>
<tr>
<td>4.4</td>
<td>( (b^n \times b^n) ) Delta Networks</td>
<td>180</td>
</tr>
<tr>
<td>4.5</td>
<td>Pruned and Augmented Delta Networks</td>
<td>184</td>
</tr>
<tr>
<td>4.6</td>
<td>Chapter Summary</td>
<td>200</td>
</tr>
<tr>
<td>5.1</td>
<td>Introduction</td>
<td>204</td>
</tr>
<tr>
<td>5.2</td>
<td>The Basic Model</td>
<td>208</td>
</tr>
<tr>
<td>5.3</td>
<td>Deadlocks</td>
<td>212</td>
</tr>
<tr>
<td>5.4</td>
<td>Performance</td>
<td>226</td>
</tr>
<tr>
<td>5.5</td>
<td>Chapter Summary</td>
<td>233</td>
</tr>
<tr>
<td>6.1</td>
<td>Summary</td>
<td>238</td>
</tr>
<tr>
<td>6.2</td>
<td>Future research</td>
<td>242</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>244</td>
<td></td>
</tr>
<tr>
<td>APPENDIX</td>
<td>249</td>
<td></td>
</tr>
</tbody>
</table>
1.1. Introduction

Packet switching refers to a method of communication used to pass messages between several interconnected systems. These messages are organized as relatively small packets that are generated by one system and passed to another. This technique has been widely applied in computer networks where the modules are geographically separated. This thesis considers the use of packet switching techniques in a class of high bandwidth interconnection networks, called delta networks, and in networks of related topology. These networks can be used to connect the modules of a modular computing system. Typical applications are in memory processor interconnections and in interprocessor communication.

A modular computing system consists of a large number of communicating processing elements or functional units, memory modules, and devices. One possible modular system organization is shown in fig. 1.1.1a. Here, any module can communicate directly with any other module by sending packets through the network. Another commonly encountered organization is shown in fig. 1.1.1b. In this case, the modules are divided into two disjoint groups connected by a network. Packets can only be passed between modules in different groups. For example,
Figure 1.1.1  Modular System Organizations
the modules in one group might be processors and those in the other group, memories.

Delta networks are composed of simple switches, called basic switches, connected together to form a large network that can provide the communication paths between the modules of modular computing systems. The networks are formed by arranging switches into stages with data paths, called links, connecting output terminals of switches in one stage to input terminals of switches in the next, adjacent stage. All links must be between switches in adjacent stages. Moreover, the pattern of links between two stages must be such that each link connects exactly one switch output terminal to exactly one switch input terminal, and there are no unconnected terminals. Input terminals of the switches in the first stage are called network input links, and output terminals of switches in the last stage are called network output links. Such networks are called multistage interconnection networks.

Information is passed through these networks in the form of small packets. In addition to a message, each packet contains information that identifies its destination module and this information is used by the switches in the networks to route the packet to that module. Thus, control of the movement of packets through the network is implemented locally in the switches, and a global controller is not needed. This is referred to as local control of packet routing. This is in contrast to circuit switched networks which first determine how to route information between two modules (i.e., select a path through the network) and then
set up a fixed connection between them by controlling the switches. The connection remains until the external controller breaks it to establish other connections.

This thesis studies the topology and the performance of these networks. The main goal of this thesis is to study how the performance of these networks is affected by changes in parameters that define their structure, size, capacity, and mode of operation. The performance of these networks is characterized in terms of how much information can be passed through them in a given period of time and how long it takes for a typical packet to pass through them. To this end, a network's throughput is defined informally as the number of packets that it can pass per unit time for a particular environment. The turn around time is the delay that a packet encounters in passing through the network. The maximum throughput that a network can deliver will be called the network bandwidth. Because several packets can be concurrently passing through a network and because packets can collide at an internal switch, the delay is not equal to the reciprocal of the throughput in general. In this thesis, the networks are modelled using Timed Petri nets [Ramachandani74a], analyzed by Markov chains [Parzen60a], and simulated with event driven simulations [Stone75a]. Bounds on the performance are derived, approximate analyses are presented, and these are substantiated by and extended to analytically intractable cases by simulation.

The basic switches in a delta network can be busses, crossbar switches, or other switch types. Thus, the full crossbar switch and the
bus are special extreme cases of delta networks. A crossbar switch with 
N input links and N output links, referred to as an (N x N) crossbar 
switch, requires O(N^2) gates in any implementation [Kuck78a], and has an 
ideal bandwidth of O(N). A single bus, on the other hand, has O(N) 
gates and O(1) ideal bandwidth. We concentrate on (N x N) delta net-
works, constructed from fixed size basic switches, which have O(N log N) 
gates and O(N) ideal bandwidth. We show that, even in realistic 
environments, their performance is comparable to or better than the per-
formance of the unbuffered crossbar switch.

Delta networks support unidirectional local control of packet rout-
ing from the network input links to the network output links. Bidirec-
tional delta networks can support local control of packet routing in 
both directions. However, there is a unique path from each network 
input link to each network output link. In fact a large number of such 
paths pass through the same switch. This leads to poor reliability 
since the failure of a single switch can break a large number of poten-
tial connections. The networks can be augmented to provide multiple 
disjoint paths from any network input link to any network output link. 
(N x N) delta networks can be pruned to obtain networks with differing 
numbers of network input and output links. Feedback paths can be 
inserted to provide a range of performance for a varying gate complex-
ity.

These networks support concurrent communication between modules. 
Several paths through the network can be active simultaneously. If data
buffers (i.e., registers capable of holding one or more packets) are added to the data paths between switches, several packets can be in various stages of passage through the network along the same path. Thus, the buffers increase the capacity of the network to pass information in exactly the same way the registers between stages of a pipelined arithmetic unit increase its computation rate [Jump78a]. Augmented delta networks have a slightly worse throughput and turn around time than delta networks under normal (fault free) operation. They can sustain communication between each input and each output link, with some degradation in performance, even with a failure of an internal switch in the network. Optimal pruned delta networks with buffers between stages have a non-intuitive topology. Inserting feedback paths leads to \((N \times N)\) networks with the gate complexity varying between \(O(N)\) and \(O(N \log N)\) and an ideal bandwidth between \(O\left(\frac{N}{\log N}\right)\) and \(O(N)\). However packets can be deadlocked in these networks. We study deadlock detection, avoidance, recovery and the performance of these networks.

1.2. Background and Related Work

Motivation

Several factors have motivated the current interest in modular computing systems organized as in fig. 1.1.1. The chief factors can be broadly classified as being technological, for improved system performance, extensibility and reliability. Among the technological reasons is the advent of VLSI which has led to new horizons in the complexity and
size of feasible systems. Current design methodologies for designing such complex systems employ top down design with structuring and modularity and result in systems composed of several communicating subsystems. The proliferation and low cost of microprocessors has spawned interest in augmenting performance by constructing multi-microprocessor systems. On the other hand, with component speeds approaching the fundamental limitations due to propagation delays, speed improvement by parallel processing has gained popularity and this has provided further impetus for the development of multiprocessor systems. These systems can be designed to allow for extensibility in functionality or performance. They can exhibit graceful degradation of performance with component failure leading to improved reliability and availability.

If these systems are to exhibit the properties of good performance, extensibility, and fault tolerance, then so must the interconnection networks. We will see that delta networks allow a large parallelism in packet transfer leading to good performance. They are extensible in that they can be constructed recursively from smaller networks. They can be augmented to exhibit graceful degradation with improved reliability and availability.

Packet Switching

There are several choices in the mode of communication between system modules. A fundamental choice is between packet switching and circuit switching [Metcalf73a]. In pure circuit switching a continuous
path between the communicating modules is established in the interconnection network and the constituent circuits are dedicated to this communication. In most networks that interconnect a large number of modules, the time required to set up a communication path is significant. Efficiency dictates that a large volume of data be passed over a circuit that has been established. Furthermore, this data should be supplied at a uniform rate close to the maximum rate that the circuit can handle. Circuit switching is inefficient for short bursty messages.

Packet switching, on the other hand, requires the inclusion of additional routing information with each packet. It is thus inefficient for long messages. However, it allows data paths to be shared leading to shorter average delays. It is possible to route packets "on the fly". System throughput can also be improved by buffering within the interconnection networks. Packet switching is particularly advantageous for short bursty messages.

There is a spectrum of communication disciplines between pure circuit switching and pure packet switching. The bandwidth of a circuit switched path can be shared by time or frequency division multiplexing to improve average delays of messages.

Computer architectures with packet communication as the basis have been proposed. A packet communication architecture system [Dennis75a] consists of a number of independent modules which communicate asynchronously by sending fixed size packets.
All communication with a module in a packet communication architecture system is in the form of packets except for communication of the initial state of the module which can be in non packet form. Each module in the system has a set of input ports over which it receives packets and a set of output ports over which it sends packets. The ports of a module receive or send packets over first in first out data channels. Module input ports receive packets from the output ports of another module or from an external source. In the latter case the input port is referred to as a system input port. Similarly, output ports either connect to input ports of another system module or they are system output ports which pass packets to destinations external to the system.

The system modules operate autonomously. This means that there is no global controller in the system. A module can thus operate on data it has received regardless of the state of other components of the system. Furthermore no timing constraints are imposed on the system modules. The modules must be designed to operate correctly regardless of the speed of channels or of the other system modules. The system is said to be speed independent. This property requires the use of an asynchronous communication protocol.

Some of the advantages of packet communication architectures are as follows. First, the modules in the system can operate concurrently, leading to parallelism and high computation rates. In conventional systems the central controller is often the bottleneck. The autonomous
operation of modules in a packet communication architecture results in no inherent bottleneck in the system.

Systems with a packet communication architecture can be designed modularly. The functional requirements of each module can be specified along with some communication standards. The individual modules can then be designed individually. Restricting the communication between modules to packets leads to simple and clean interfaces between system modules. The absence of timing restriction leads to simple functional specification of the modules.

Proving the correctness of a packet communication architecture system is relatively easy since this proof can be made modular. That is, the correct operation of the entire system can be proved by proving the correctness of each module and then showing that the system will operate correctly if each of the modules operate correctly. This is in contrast to systems with a shared memory or with interrupts.

The speed independence of the modules in the system is an asset in tuning a designed system for better performance. Bottlenecks in the system can be identified and redesigned to operate faster without affecting the correctness of the operation of the rest of the system.

Interconnection networks for packet communication architectures satisfy the requirements and have the advantages listed above. The networks that we study in this thesis are examples of such networks. However, they can also be used in systems that do not satisfy some of the
assumptions of packet communication architectures. For instance, they can be used in synchronous systems. The performance results in this thesis are applicable to such systems as well.

**Interconnection networks**

Most of the previous work on interconnection networks for modular systems has been concerned with networks that do not use packet switching techniques and do not contain buffers. The switches are usually controlled by an external controller instead of by information in packets. Much of this work has concentrated on the use of networks to connect each of the input ports to exactly one output port. Then, packets are passed from all of the input ports to all of the output ports concurrently. In this mode of operation, the state of the network can be described as a permutation \( \Pi \) on the set \( \{1, 2, \ldots, n\} \), where input port \( i \) is connected to output port \( \Pi(i) \). Networks operated this way are called permutation networks. The goal of much of the research on permutation networks is to characterize the class of permutations that can be realized using a particular interconnection pattern for the links between the stages, and to develop algorithms which can take advantage of the structure of the permutation networks.

The interconnection structure of many well-known permutation networks satisfy the properties required by delta networks. For example, the OMEGA network [Lawrie75a], the indirect binary \( n \)-cube [Pease75a], the cube network [Siegel78a], the flip network [Batcher76a], the regular
SW banyan network [Goke73a], the baseline network [Wu80a], and the reverse exchange network [Wu80b] are all instances of delta networks. However, when delta networks are operated as packet switching networks, their performance, measured by throughput and delay, is essentially independent of which interconnection structure is used, as we will see later in this thesis. Hence, very little of the previous analytical work on permutation networks can be used to analyze the performance of packet switching in these networks.

Another area that has been investigated extensively is that of telephone switching networks [Benes65a]. Crossbar switches were first used in these networks. More recently, circuit switched crossbar switches have been proposed and studied for multiprocessor interconnection networks in [Wulf72a; Franklin79a]. An \( (N \times N) \) crossbar switch requires \( O(N^2) \) gates while a lower bound on the number of gates required to allow all permutations is \( O(N \log N) \) [Kuck78a]. Optimal rearrangeable multistage networks that achieve this lower bound were proposed in [Benes64a] and are referred to as Benes networks. These networks have now found application as permutation networks. One of the goals of current research in these networks concerns efficient means of controlling Benes networks [Parker80a; Nassimi79a; Nassimi80a; Chow80a]. These networks have also been studied extensively in a circuit switching environment. The chief performance criterion used is the blocking probability which is the probability of being unable to complete a circuit between the desired points. The techniques used and the results obtained for the circuit switching environment do not carry over to
packet switching.

There has been less research on the use of packet switching networks to interconnect the modules of a modular system. The data flow architecture currently under development at the Massachusetts Institute of Technology will use buffered networks to interconnect the processor modules with the memory modules [Misunas78a; Dennis80a]. These networks will operate asynchronously. Some initial analysis of their performance can be found in [Misunas75a; Jacobsen77a; Boughton79a]. The TRAC system under construction at the University of Texas in Austin will use a synchronous packet switching network with a single level of buffering between stages [Tripathi79a]. An orthogonal bus interconnection structure has been proposed and analysed in [Gonsalves79a]. Hierarchical bus structures have been proposed in [Swan77a; Wittie76a; Wu80c], and a tree structure in [Despain78a]. However, only loose bounds on the performance of these networks have been derived.

The class of delta networks was defined in [Patel79a]. He analyzed the performance of unbuffered delta networks and showed that their performance was independent of the choice for the interconnection pattern. An unbuffered delta network has been proposed as a multiprocessor memory interconnection and studied by simulation in [Barnes80a; Lundstrom80a]. In this thesis we extend some of Patel's results on unbuffered delta networks and investigate how the addition of buffers to the links between stages affects the networks performance. We also consider augmenting these networks for better reliability, pruning them to have
different numbers of input and output links, and inserting feedback
paths to obtain a range of performance. Parts of this study have been
reported in [Dias81a; Dias80a; Dias81b].

Most of the research on packet switching networks has been for com-
puter networks used to interconnect computers at geographically
separated sites [Blanc80a]. There are significant differences in the
way these networks are designed and used as compared to the interconnec-
tion networks described above. For example, there are no intermediate
nodes in most computer networks as there are in the inner stages of an
interconnection network. The bandwidth of the channels between nodes,
instead of the delay at nodes, is the main limit to performance in com-
puter networks. Also, the dominant cost of a computer network is often
the cost of the channels instead of the nodes. Finally, the nodes of a
computer network are quite complex (usually general purpose computers)
and capable of realizing complex routing strategies.

Recently, there has been a great deal of interest in local networks
of computers [Thurber80a]. They provide a degree of coupling between
system modules that is somewhere between large computer networks and
interconnection networks. However, the way they are used and designed
differs from interconnection networks in much the same way as computer
networks.
Network Classification and Characteristics

We now look at a classification of interconnection networks. A taxonomy for interconnection networks has been given in [Anderson75a]. There are three main levels in this taxonomy. The first is a choice between direct or indirect transmission of packets from source to destination and is referred to as transfer strategy. The second relates to the control of the transfer and is either centralized or decentralized control. At the third level is the choice between shared or dedicated packet transfer paths. In terms of this taxonomy the networks we consider use indirect transmission with decentralized control and shared paths. The use of indirect instead of direct communication between modules removes the necessity of having a direct connection between all modules. Direct connections between N modules would require O(N^2) gates. Having decentralized (variously referred to as distributed or local) control of network traffic prevents a potential bottleneck that could occur at a central controller. Sharing of paths leads to a reduction in cost. Indirect communication and sharing of paths allow alternate routing of packets in case of a failure. We now list some network characteristics that result from these and further implementation decisions [Anderson75a; Thurber74a].

Cost: The cost of an interconnection network depends intimately on the precise implementation including the technology used. The cost of the network is difficult to estimate, especially if a cost estimate is required before implementation decisions are made. Some facets of
networks on which the cost depends are as follows. If the networks is made up of similar 'basic cells' then the number of cells and the complexity of each such cell are indicators of cost. The asymptotic gate complexity is another useful measure. The number of interconnecting wires between basic cells is a is important, particularly if the basic cells are assumed to be available as VLSI or LSI chips. If the interconnection network or part of it is to be implemented in VLSI then the area that is required on a chip is a cost indicator.

**Extensibility:** Extensibility is a measure of the ease with which the network can be expanded to connect a larger number of modules. There are several factors which affect this property. The **cost modularity** is the incremental cost of adding extra (input or output) links. The **place modularity** relates to whether there are any restrictions to where additional modules can be connected. The **connection flexibility** for indirect communication relates to the number of additional elements that need to be added to the interconnection network to accommodate an additional input or output element.

**Thruput and Bandwidth:** The thruput for interconnection networks in a packet communication environment can be defined as the number of packets that the network puts out in unit time for a particular environment. Thus, this criterion depends on the specific environment. The maximum thruput that the network can support is referred to as the **network bandwidth**. A network characteristic of interest is its thruput under extreme environmental conditions. An upper extreme is one in
which packets are always assumed to be available at network inputs whenever the network can accept them. Bounds on the network throughput are also of interest. Other throughput related characteristics are as follows. The **throughput extensibility** relates to variation in throughput with size of the network. It indicates whether there are fundamental bounds on the throughput. The **throughput stability** relates to how the network behaves with increasing load. The network is said to be throughput stable if the throughput increases monotonically with increasing network load. This property indicates whether or not packet input flow control is required or desirable.

**Turn around time:** The delay that a packet encounters in passing through the network is called the turn around time. As with the throughput, this is environment dependent. Of interest are bounds on the turn around time and the turn around time for extreme environments. The **turn around time stability** indicates whether this performance criterion becomes unacceptably large at high loads.

**Reliability:** This property concerns the network behavior under faults. The **failure effect** is a measure of the fraction of the system communication that is halted by the failure of a component. The **failure reconfiguration** relates to the complexity of network reconfiguration to accommodate a fault.

**Network control:** We have already referred to the issue of distributed (local or decentralized) versus global (or centralized) control of packet routing through the network and the potential bottleneck that the
latter option entails. There is a spectrum of choices between these extremes. For instance, the network could be partitioned into sub-networks with a single controller for each partition, as in multistage interconnection networks with individual stage control [Siegel77a]. Another aspect of network control is the complexity of the packet routing algorithms.

**Partitioning:** This property reflects the ability of the network to partition its inputs and outputs into partitions with no communication across partitions.

**Bottlenecks:** This relates to the uniformity of packet load on the components of the network.

There are a host of other practical issues. Examples are: the development of LSI/VLSI partitions of the network that can be used as basic building blocks and the gate to pin ratios of these partitions; the ability to use off-the-shelf parts for construction; the ease of reduction to design, and so on.

The characteristics of delta and related networks in these terms will be discussed later in the thesis, as the properties are exposed.

1.3. **Plan of the Thesis**

In chapter 2 of this thesis we study the topology of delta networks, bidirectional delta networks, augmented bidirectional delta networks and pruned delta networks. We specify construction algorithms
and develop network properties that are used in the analyses of later chapters.

Chapter 3 deals with the performance of unbuffered delta networks (i.e., of delta networks with no internal buffers). It is shown how a simple model for network operation can be used to obtain a 'network characteristic' that can be used to effectively estimate the performance for other models and environments. Simulation is used to validate the analysis and to extend it to analytically intractable models.

Chapter 4 is devoted to the study of buffered networks. This study reveals that the networks are throughput stable and that a considerable improvement in performance is obtained by buffering in the network. The performance obtained is found to be sensitive to precisely how the switches in the network operate. The performance of buffered pruned and augmented delta networks is also studied. Some non-intuitive optimal pruned networks are obtained.

Chapter 5 studies the implications of adding feedback paths to these networks. It is shown that deadlocks can arise. The aspects studied are those of deadlock detection, avoidance, and recovery and the performance of these networks. It is shown that these networks are not throughput stable and that some packet input flow control is necessary to obtain reasonable performance.

Finally, chapter 6 summarizes the results of the thesis, and suggests areas for future research.
CHAPTER 2

Delta and Related Networks

Topology and Properties

2.1. Introduction

In this chapter we define the topology and derive topological properties of delta networks, bidirectional delta networks, augmented delta networks and regular pruned delta networks. The properties derived here form the basis of the performance analyses of the following chapters.

The primary goals underlying the definition and the evolution of these networks are those of good performance, local (or distributed) control, extensibility, and reliability. Thus, we would like to obtain a performance comparable to or better than that of the crossbar switch, but with fewer gates than the $O(N^2)$ required by an $(N \times N)$ crossbar switch. Further, a uniformity in the packet traffic at different switches in the network is desirable. This eliminates the possibility of bottlenecks due to uneven loading at switches. Using local control in packet routing prevents the possible bottleneck and the reliability problem of a central controller. Using a simple routing strategy minimizes the routing overhead in terms of cost and performance. The networks should be capable of connecting a large number of modules in a modular computing system, and should be extensible to allow the addition
of further modules to the system. Finally, the networks should be able to tolerate internal switch failures.

The performance requirement is met by delta networks constructed from fixed size basic switches. \((N \times N)\) delta networks have \(O(N \log N)\) gates and \(O(N)\) ideal bandwidth. A packet is routed through these networks by examining different digits of the destination address of the packet at different stages in the network and directing it to a corresponding switch output link. Thus, packet routing is very simple and efficient. Bidirectional delta networks allow digit controlled routing in both directions (i.e., from input to output links and vice versa). The networks can be constructed recursively from smaller networks and are thus extensible. However, there is a unique path from each network input link to each network output link leading to poor reliability. This problem is alleviated by adding an additional stage to obtain augmented delta networks. These networks operate with degraded performance under internal switch failures. Finally, \((N \times N)\) delta networks can be pruned to have different numbers of input and output links. We discuss the relative merits of several ways of doing this. Quantitative performance results are derived in the following chapters.
2.2. Delta networks and their properties

In this section we define delta networks, present some of their properties and their implications, and specify some methods for constructing them.

The name "delta networks" was introduced by Patel [Patel79a]. We consider \((b^n \times b^n)\) delta networks, for positive integers \(b\) and \(n\). The general form of these networks is shown in fig. 2.2.1. These networks have \(b^n\) network input links and \(b^n\) network output links, each labelled by \(n\)-digit base \(b\) numbers. They are composed of \(n\) stages \(S_0, S_1, \ldots, S_{n-1}\). Each stage has \(b^{n-1}\) \((b \times b)\) "basic switches" labelled \(0, 1, \ldots, (b^{n-1}-1)\). There are permutations \(P_0, P_1, \ldots, P_{n-2}\) between stages. The outputs of the \((b \times b)\) basic switches are labelled \(0, 1, \ldots, (b-1)\). These basic switches should be able to connect any of their input links to any of their output links. (It is not necessary that these connections can be made simultaneously.)

The permutations between stages must satisfy the requirement that there exists a transmission path (hereafter referred to as a path) between any network input link \(i_{n-1} \ldots i_0\) and any network output link \(d_{n-1} \ldots d_0\) obtained by connecting input links of switches in stage \(j\), \(0 \leq j < n\), to output \(d_{c_j}\), \(0 \leq c_j < n\). This property of delta networks is called the digit controlled property. We refer to \(d_{c_j}\) as the control digit at stage \(j\) while \(c_j\) is called the control index at stage \(j\).
Figure 2.2.1  \((b^n \times b^n)\) delta networks.
The immediate question that arises is whether any networks satisfying these properties exist. Several special cases of delta networks have been proposed in the literature and hence the class is not empty. For example, the OMEGA network [Lawrie75a], the indirect binary n-cube [Pease75a], the cube network [Siegel78a], the flip network [Batcher76a], the baseline network [Wu80a], and the reverse exchange network [Wu80b], are all instances of delta networks (except possibly for a permutation at either the input or at the output of these networks). Fig. 2.2.2 shows a \( (2^3 \times 2^3) \) delta network with a perfect shuffle permutation [Stone71a] between the stages of the network. This can be generalized to a \( (b^n \times b^n) \) network. We will specify formal procedures for constructing delta networks after some of their properties have been exposed.

**Properties of delta networks**

We first make some observations that follow directly from the definition of these networks. One of the main ideas behind the definition of these networks is that of allowing simple local control in routing packets. This is based on the digit controlled property, defined above. Each stage in the network need only examine a different bit of the destination address of a packet and direct it to the switch output link bearing that label. The control index at that stage specifies which bit of the destination address to look at. An example of routing a packet through a network is shown in fig. 2.2.2. Suppose that a packet arrives at network input link 101 and is directed to network
Figure 2.2.2: Example of a delta network (OMEGA network without input permutation).
output link 011. Then, the packet is routed through the network as indicated by the dashed line. At stage $S_0$, the control index is 2, and bit $d_2$ of the packet destination address is 0. Thus, the packet is routed to the switch output link with label 0, as shown in the figure. A similar routing strategy is used at the other stages of the network. The digit controlled property of these networks ensures the correctness of this routing algorithm.

Delta networks are composed of interconnected basic switches. These basic switches have not been specified in the definition of delta networks. All that is required of these switches is that they should be able to connect each of their input links to each of their output links and these connections need not be simultaneous. One extreme case is when the basic switches are crossbar switches. For this case, input links can be simultaneously connected to output links, provided no two inputs are to be connected simultaneously to the same output link. The other extreme case is when the basic switches are similar to a bus. For this case at most one input link can be connected to an output link at any instant of time. In fact, the basic switch can itself be a delta network with smaller basic switches. This possibility will be examined in a later section.

Consider $(N \times N)$ delta networks with $N = b^n$, where $b$ is the size of the basic switches in the delta network as defined above, for $1 \leq n \leq \log_b N$, and $2 \leq b \leq N$. As discussed above, the extreme cases are the use of bus type and crossbar type basic switches. A $(b \times b)$
crossbar switch has $O(b^2)$ gates, while a $(b \times b)$ bus type switch has $O(b)$ gates [Kuck78a]. Further, there are $\frac{N}{b} \log_b N$ $(b \times b)$ basic switches in the network. Thus, a delta network with crossbar type basic switches has a gate complexity of

$$O(b^2 \cdot \frac{N}{b} \log_b N) = O(\frac{N \cdot b \log N}{\log b})$$

while that of a delta network with bus type basic switches is,

$$O(b \cdot \frac{N}{b} \log_b N) = O(\frac{N \cdot \log N}{\log b})$$

Another measure of interest is the total number of links between the stages of the network. There are $\log_b N$ stages and $N$ links between stages leading to a total of $N(\log_b N - 1)$ or $O(\frac{N \log N}{\log b})$ links between stages. This measure is of interest if the basic switches are already available, say as VLSI chips [Mead80a]

Some cases of interest are:

Fixed $b$: For this case, the gate complexity and the number of links between stages are both $O(N \log N)$ for both crossbar and bus type basic switches. This is the most common case envisaged. All the examples of delta networks in the literature, referred to above, are for the case of $b = 2$. This case is also of interest in building large networks from small basic
switches.

Fixed N: For the case of crossbar type basic switches, the gate complexity is $O(\frac{b}{\log b})$. The use of bus type basic switches results in a gate complexity of $O(\frac{1}{\log b})$. Thus, for constant N, using crossbar type switches results in an increase in asymptotic gate complexity with an increase in the basic switch size, b, while the use of bus type switches results in the asymptotic gate complexity decreasing with an increase in b. The number of links between stages is $O(\frac{1}{\log b})$, resulting in a decreasing number of links with an increase in b.

There are other parameters of interest depending on the application. For instance, if the entire interconnection network is to be built on a single chip, then the area occupied by the network on the chip is important and developing bounds on this parameter is an area of current research [Leiserson80a; Franklin80a].

We now present some properties that follow as a consequence of the definitions of these networks. The definition of delta networks requires the existence of a path from each network input link to each network output link that satisfies the digit controlled property. The following lemma shows that each such path is unique.

**Lemma 2.2.1** In a $(b^n \times b^n)$ delta network there is a unique path from each network input link to each network output link.
Proof. There are $b^n$ paths from each network input link to network output links (since there are $n$ stages and $b$ positions for switch connections in each stage). But in a $(b^n \times b^n)$ delta network, there must be at least one path to each of the $b^n$ output links from any given network input link and hence the paths must be unique.

This property means that the reliability of these networks is poor. It indicates that communication between a network input link and a network output link is broken if any of the switches on the unique path between these links fail. In fact, the reliability situation is even worse as the following lemma indicates.

**Lemma 2.2.2** Consider any switch "m" in stage $k$ of a $(b^n \times b^n)$ delta network, for $1 \leq k < n$. Let $i_1, \ldots, i_b$ ($o_1, \ldots, o_b$) denote the input (output) links of m. Let $I_1, \ldots, I_b$ ($O_1, \ldots, O_b$) be the sets of network input (output) links with paths to (from) $i_1, \ldots, i_b$ ($o_1, \ldots, o_b$) respectively. (See fig. 2.2.3.) Then,

(i) The sets $I_1, \ldots, I_b$ ($O_1, \ldots, O_b$) are disjoint.

(ii) $|I_1| = \ldots = |I_b| = b^k$.

(iii) $|O_1| = \ldots = |O_b| = b^{n-1-k}$.
Illustration for Lemma 2.2.2

Stage $S_k$, $0 \leq k < n$
(iv) A path from any network input link in $I_1 U \ldots U I_b$ to any network output link in $O_1 U \ldots U O_b$ must pass through switch $m$.

(v) There are $b^{n+1}$ such paths.

**Proof** Suppose that statement $\text{(i)}$ is false. Then there is more than one path from some input link to switch $m$, and thus from some input link, to any output link in $O_1 U \ldots U O_b$, contradicting lemma 2.2.1. This proves $\text{(i)}$.

Statement $\text{(ii)}$ follows by an easy induction on $k$, while $\text{(iii)}$ follows by a backward induction on $k$. Statement $\text{(iv)}$ is a consequence of $\text{(i)}$ and lemma 2.2.1, while $\text{(v)}$ follows trivially from $\text{(ii)}$, $\text{(iii)}$ and $\text{(iv)}$.

Statement $\text{(i)}$ of the above lemma is the basis of some of the approximations in the succeeding chapters. Statements $\text{(ii)}$ through $\text{(v)}$ indicate that $b^{n+1}$ paths, from network input links to network output links, pass through each switch in the network. This means that the failure of a single switch cuts off that number of communication paths, leading to poor network reliability. This is precisely the problem that the "augmented delta networks" alleviate, as we will see in a later section. A positive aspect of this property is that each switch in the network has the same number of paths from network input links to network output links passing through it, leading to a uniformity in the packet
traffic at different switches.

We now consider some properties which can be used to construct delta networks.

**Lemma 2.2.3** Consider any path from a network input link to a network output link labelled $d_{n-1} \ldots d_0$ in a $(b^n \times b^n)$ delta network. Suppose that this path includes a switch output link labelled $j$, $0 \leq j < b$, in stage $k$, $0 \leq k < n$, with control index $c_k$. Then $d_{c_k} = j$.

**Proof** Follows directly from the uniqueness of paths from network input links to network output links (lemma 2.2.1) and the digit controlled property of delta networks.

**Lemma 2.2.4** Consider any switch "$m$" in a $(b^n \times b^n)$ delta network that is not in stage $S_0$. Let $m_1, \ldots, m_b$ be the predecessor switches of $m$. Then the labels of output links, $l_1, \ldots, l_b$ of $m_1, \ldots, m_b$ respectively, that are connected to $m$, are identical.

**Proof** Follows directly from lemma 2.2.3.

**Lemma 2.2.5** Consider a $(b^n \times b^n)$ delta network $A$. Suppose that $\beta : \{0, \ldots, n-1\} \rightarrow \{0, \ldots, n-1\}$ is a bijection. Let network $A^*$ be obtained from network $A$ by changing the control index at stage $k$,
0 \leq k < n$, from $c_k$ to $\beta(c_k)$, and the label of any network output link from $d_{n-1} \cdots d_0$ to $d_{n-1}^* \cdots d_0^*$, such that $d_{\beta(c_k)} = d_{c_k}$. Then network $A^*$ is also a delta network.

Proof Follows easily from lemma 2.2.3. 

**Theorem 2.2.1** Splitting a $(b^n \times b^n)$ delta network into $b$ $(b^{n-1} \times b^{n-1})$ delta networks.

Let $A$ be a $(b^n \times b^n)$ delta network. Let $A^k$, $0 \leq k < b$ be obtained from $A$ as follows (see fig. 2.2.4).

(i) Delete all switches in stages $S_1, \ldots, S_{n-1}$ that are not on any paths from network input links to network output links of $A$ that pass through switch output links of $S_0$ labelled $k$.

(ii) Delete stage $S_0$. Consider input links of switches in stage $S_1$ as network input links. Arbitrarily label these new network input links by $(n-1)$ digit base $b$ numbers.

(iii) Relabel $S_1$ as $S_0$, $\ldots$, $S_{n-1}$ as $S_{n-2}$. Arbitrarily relabel the remaining switches in each stage as $0, 1, \ldots, (b^{n-2} - 1)$. 
Network $A : (b^n \times b^n)$ delta network

Stage $S_0$

* Figure 2.2.4 Illustration for theorem 2.2.1
For any network output link labelled as $d_{n-1} \ldots d_0$, delete bit $d_{c_0}$ to obtain an $(n-1)$ bit label. Let the control index at stage $j$ of $A^k$ be $c_{j+1}$ of network $A$, if $c_{j+1} < c_0$, or $(c_{j+1} - 1)$ of network $A$, otherwise.

Then $A^k$ is a $(b^{n-1} \times b^{n-1})$ delta network. Further, the subnetworks $A^0, \ldots, A^{b-1}$ are disjoint (i.e. there is no switch in $A$ that belongs to more than one of $A^0, \ldots, A^{b-1}$).

Proof It easily follows from lemma 2.2.3 that the subnetworks $A^0, \ldots, A^{b-1}$ are disjoint and that each satisfies the digit controlled property. The other relabelling steps indicated merely make $A^k, 0 \leq k \leq b$, satisfy the definitions of a delta network.

We now examine the possibility of replacing the basic switches in the network by delta networks with smaller size basic switches. We show that, under certain constraints, the networks that result are themselves delta networks. An example of a $(4^2 \times 4^2)$ delta network is shown in fig. 2.2.5. The basic $(4 \times 4)$ switches in this network are replaced by $(2^2 \times 2^2)$ delta networks resulting in the $(2^4 \times 2^4)$ delta network shown in the figure. The following algorithm specifies how this sort of basic switch replacement can be done.

Algorithm 2.2.1. Replacing the basic switches of a delta
Figure 2.2.5  Replacing the basic (4 x 4) switches in a (4^2 x 4^2) delta network by (2^2 x 2^2) delta networks.
network by sub delta networks to obtain a delta network with smaller basic switch size.

Given a \((b^n \times b^n)\) delta network "A" with \(b = b_1^m\), the \((b \times b)\) switches in network A are replaced by \((b_1^m \times b_1^m)\) delta networks to obtain a \((b_1^{nm} \times b_1^{nm})\) delta network \(A^*\), as follows.

(i) For all \((b_1^m \times b_1^m)\) delta networks that replace \((b \times b)\) switches in stage \(S_k\) of network A, \(0 \leq k < n\), the control index at any stage of these (sub) networks must be the same. (Lemma 2.2.5 can be used to enforce this.) Suppose that the control index at stage \(S_k\) of A is \(c_k^*\), and that the control index at stage \(S_j\) of the sub-networks that replace switches in stage \(S_k\) of A is \(c_j^*\), for \(0 \leq j < m\). Then, the control index of that stage of network \(A^*\) is set to \((m \cdot c_k^* + c_j^*)\).

(ii) Any two output links of \((b \times b)\) switches in stage \(S_k\) of network A, \(0 \leq k < n\), that have the same label are given the same label after the replacement of the \((b \times b)\) switches by \((b_1^m \times b_1^m)\) delta networks. That is, define a bijection,

\[ p_k : \{0, \ldots, b\} \rightarrow \{(0 \ldots 0), \ldots, (b_1-1 \ldots b_1-1)\} \]

from the labels of output links of \((b \times b)\) switches of A to
the \( m \)-digit base \( b \) labels of sub-network output links.

(iii) Suppose that network \( A \) has \( c_k \) as the control index at stage \( k \), \( 0 \leq k < n \). Consider any network output link of \( A \) labelled \( d_{n-1} \ldots d_0 \). Then, the new label of that output link in \( A^* \) is obtained by replacing each base \( b \) digit \( d_{c_j} \), \( 0 \leq j < n \), by the \( m \)-digit base \( b_1 \) number \( b_j(d_{c_j}) \).

**Theorem 2.2.2**  For any \( (b^n \times b^n) \) delta network \( A \) with a basic switch size of \( b = b_1^m \), algorithm 2.2.1 constructs a \( (b_1^{n+m} \times b_1^{n+m}) \) delta network \( A^* \) with basic switch size of \( b_1 \).

**Proof**  The network \( A^* \) has \( b_1^{nm} \) network input and output links and has \( (n \times m) \) stages. It also has a path from each network input link to each network output link (since network \( A \) is a delta network and the basic switches are replaced by delta networks). The digit controlled property of \( A^* \) follows easily from the construction and the fact that \( A \) has the digit controlled property.

\[ \square \]

This construction procedure can be used recursively to construct some \( (b^m_2 \times b^m_2) \) delta networks. Fig. 2.2.5 is an example of constructing a \( (2^2 \times 2^2) \) delta network using this procedure recursively. We start with a single \( (2^2 \times 2^2) \) switch. This is constructed in terms of
\( (2^{2^{-1}} \times 2^{2^{-1}}) \) switches. These switches are in turn replaced by \( (2^0 \times 2^0) \) size switches. However, all delta networks cannot be constructed using this method. This theorem will be used in analyzing the performance of \( (b^n \times b^n) \) buffered delta networks in chapter 4.

**Constructing Delta Networks.**

Delta networks can be constructed recursively using theorem 2.2.1. This theorem implies that any \( (b^n \times b^n) \) delta network, for \( n > 1 \), can be constructed from two \( (b^{n-1} \times b^{n-1}) \) delta networks, as indicated in fig. 2.2.4. The boundary condition is for \( n = 1 \) and this corresponds to a single \( (b \times b) \) switch. The construction can be formalized as follows.

**Algorithm 2.2.2** Constructing a \( (b^{n+1} \times b^{n+1}) \) delta network from \( (b^n \times b^n) \) delta networks.

Let \( A^0, \ldots, A^{b-1} \) be \( (b^n \times b^n) \) delta networks with the same control index at the same stage. (The construction in lemma 2.2.5 can be used to alter these networks to satisfy this requirement.) Then, a \( (b^{n+1} \times b^{n+1}) \) delta network "A" can be obtained as follows.

1. Relabel the stages \( S_0, \ldots, S_{n-1} \) as \( S_1, \ldots, S_n \), respectively in networks \( A^0, \ldots, A^{b-1} \).
(ii) Include a new stage $S_0$ with $b^n$ $(b \times b)$ basic switches. Make connections from stage $S_0$ to $A^0$, ..., $A^{b-1}$ and label output links of switches in $S_0$ as shown in fig. 2.2.4.

(iii) Let the control index, $c_0$, at stage $S_0$ be $n$. Any output link with label $d_{n-1} \ldots d_0$ in $A_k$, $0 \leq k < b$, is relabelled by the $(n+1)$ digit label $k \ d_{n-1} \ldots d_0$.

(iv) Arbitrarily relabel the new network input links (i.e. input links of stage $S_0$ switches) by distinct $(n+1)$ digit base $b$ numbers. Arbitrarily relabel the switches at any stage $S_k$, $0 \leq k < n$, as 0, 1, ..., $(b^n-1)$.

The control indices can now be permuted, if desired, using the construction of lemma 2.2.5.

□

**Theorem 2.2.3** All $(b^n \times b^n)$ delta networks can be constructed by starting with $(b \times b)$ basic switches and using algorithm 2.2.2 repeatedly.

**Proof** By an easy induction on $n$ and using theorem 2.2.1.

□

An example of using this procedure to construct a $(2^3 \times 2^3)$ delta network is shown in fig. 2.2.6.
Figure 2.2.6 Example of the recursive construction of a delta network.
2.3. Bidirectional delta networks

Delta networks allow for simple local control in packet routing, from network input links to network output links. Bidirectional delta networks extend this to permit simple routing of packets in both directions.

A \((b^n \times b^n)\) bidirectional delta network is a \((b^n \times b^n)\) delta networks with the following additional properties.

(i) The input links of the \((b \times b)\) basic switches are also labelled \(0, 1, \ldots, (b-1)\) (in addition to such a labelling of the output links of the \((b \times b)\) basic switches in delta networks).

(ii) The network is also a delta network when the switch and network input links are considered as switch and network output links respectively. The control index at stage \(S_k\), \(0 \leq k < n\), used in routing packets from network output links to network input links, is denoted by \(c^*\), and is referred to as the reverse control index at stage \(k\). (It is not necessary that \(c_k = c^*_k\).)

We now show that bidirectional delta networks can be split into smaller bidirectional delta networks. This property is analogous to theorem 2.2.1.
Theorem 2.3.1 Splitting a \((b^n \times b^n)\) bidirectional delta network into \(b \times (b^{n-1} \times b^{n-1})\) bidirectional delta networks.

Let \(A\) be a \((b^n \times b^n)\) delta network with \(n > 1\). Let \(A^k, 0 \leq k < b\) be obtained from network \(A\) as in the statement of theorem 2.2.1 with the following differences. In step (ii) of the statement of theorem 2.2.1 do not arbitrarily label the new network input links. Instead, add the following step.

(v) For any network input link \(I\), labelled \(i_{n-1} \ldots i_0\), delete digit \(d_*\) to obtain an \((n-1)\) digit label. Associate this label \(c_0\) with the (new) network input link of \(A^k\) that is connected to \(I\) through a single switch. Let the reverse control index at stage \(j\) of \(A^k, 0 \leq j < n-1\), be \(c_j^*\) of network \(A\), if \(c_j^* \neq c_0\), or \((c_j^* - 1)\) of network \(A\), otherwise.

Then \(A^k\) is a \((b^{n-1} \times b^{n-1})\) bidirectional delta network. Further, the sub-networks \(A^0, \ldots, A^{b-1}\) are disjoint (i.e., there is no switch in \(A\) that belongs to more than one of \(A^0, \ldots, A^{b-1}\)).

Proof From Theorem 2.2.1 it immediately follows that \(A^k, 0 \leq k < b\), are disjoint delta networks. The bidirectionality of these networks follows from lemma 2.2.3 and the above construction.
Bidirectional delta networks can be constructed using the algorithm to construct delta networks with some modifications.

Algorithm 2.3.1 Constructing a \((b^{n+1} \times b^{n+1})\) bidirectional delta network from \((b^n \times b^n)\) bidirectional delta networks.

Let \(A^0, \ldots, A^{b-1}\) be \((b^n \times b^n)\) bidirectional delta networks with the same control index at the same stage. (The construction of lemma 2.2.5 can be used to alter these networks to satisfy this requirement.) Then a \((b^{n+1} \times b^{n+1})\) bidirectional delta network "A" can be obtained using algorithm 2.2.2 with the following changes.

(a) In step (ii) of algorithm 2.2.2 the labels of the network input links of \(A^0, \ldots, A^{b-1}\), to which a switch in stage \(S_0\) is connected, must be identical.

(b) In step (iv) of algorithm 2.2.2 the network input links of "A" are not labelled arbitrarily but are instead labelled as follows. Let the reverse control index, \(c^*_0\), at stage \(S_0\) be \(n\). Consider any switch \(m\) in stage \(S_0\) with an input link \(i\) labelled \(j, 0 \leq j < b\). Suppose that the labels of network input links of \(A^0, \ldots, A^{b-1}\) that are connected to \(m\) have (the same) label \(d_{n-1}, \ldots, d_0\). Then \(i\) is also a network input link of \(A\) and is labelled by the \((n+1)\) digit number \(j d_{n-1} \ldots d_0\).
As in algorithm 2.2.2, the reverse control indices can now be permuted using the construction of lemma 2.2.5.

**Theorem 2.3.2** All \((b^n \times b^n)\) bidirectional delta network with positive integer \(n\) can be constructed by starting with \((b \times b)\) basic switches and using algorithm 2.3.1 repeatedly.

**Proof** By an easy induction on \(n\) and using theorem 2.3.1.

An example of constructing a \((2^3 \times 2^3)\) bidirectional delta network is shown in fig. 2.3.1.

We now show that all bidirectional delta networks are topologically equivalent. We need the following definitions.

**Definition** Two \((b^n \times b^n)\) delta networks \(A\) and \(A'\) are said to be identical if, for identically labelled switches \(a\) in \(A\) and \(a'\) in \(A'\) in the same stage \(S_k\), \(0 \leq k < n\), the following statements are true.

(i) The labels of switches on identically labelled output links of \(a\) and \(a'\) are the same.

(ii) If \(k=n-1\) then the labels of the network output links connected to \(a\) and \(a'\) are the same.
Figure 2.3.1 A bidirectional delta network.
(iii) If \( k=0 \) then the labels of the network input links connected to \( \xi \) and \( \xi' \) are the same.

(iv) The control index at stage \( k, 0 \leq k < n \), is the same in \( A \) and \( A' \).

**Definition** Two \((b^n \times b^n)\) delta networks \(A\) and \(A'\) are said to be topologically equivalent if the switches, network input, and network output links can be renumbered, and the control indices of \(A'\) changed, to obtain a delta network \(A''\) that is identical to network \(A\).

**Theorem 2.3.3** All \((b^n \times b^n)\) bidirectional delta networks are topologically equivalent.

**Proof** Consider any two \((b^n \times b^n)\) bidirectional delta networks \(A\) and \(A'\). We show by an induction on \(n\) that the networks \(A\) and \(A'\) are equivalent.

**Basis** For \(n=1\) single \((b \times b)\) switches result and the statement is trivially true.

**Induction step** Suppose that the statement is true for \(n = k\). Consider \(n = k+1\). Then from theorem 2.3.1 networks \(A\) and \(A'\) have the structure shown in fig. 2.3.2 where \(A^0, \ldots , A^{b-1}\) and \((A^0)', \ldots , (A^{b-1})'\) are all \((b^k \times b^k)\) bidirectional delta networks. By hypothesis all these sub-networks are topologically equivalent. Hence, networks \((A^0)'', \ldots , (A^{b-1})''\) that are identical to networks \(A^0, \ldots , A^{b-1}\) respectively can be obtained from networks
Figure 2.3.2 Figure for theorem 2.3.3
$(A^0)'$, $\ldots$, $(A^{b-1})'$ respectively. Now, from lemma 2.2.3 (applied from network output to input links), it follows that the switches in stage $S_0$ of $A'$ must be connected to network input links in $(A^0)'$, $\ldots$, $(A^{b-1})'$ with the same label. (Note that the bidirectional property of the networks is used in applying the lemma form network output to network input links.) Further, this property must hold in $(A^0)'$, $\ldots$, $(A^{b-1})'$, since the only valid transformations of control indices and the labels of input and output links are of the form of lemma 2.2.5 (as required by lemma 2.2.3). Using this property the switches in stage $S_0$ of $A'$ can be uniquely relabelled and the control index at stage $S_0$ changed so that the resulting network is identical to network $A$.

2.4. **Augmented delta networks**

Delta networks have a unique path from each network input link to each network output link. This stems from the underlying $b$-ary tree with $n$ levels from a particular network input link to output links of a $(b^n \times b^n)$ delta network. We would like to have multiple disjoint paths, between any network input and output links, for better reliability. However, we want the network to retain digit controlled local routing of packets, and an $O(N \log N)$ gate complexity for an $(N \times N)$ network.

These requirements can be met by adding an additional stage to delta networks. Consider a delta network, $A$, shown in fig. 2.4.1, with
Figure 2.4.1 Constructing an augmented delta network
sub delta networks $A^0$, $\ldots$, $A^{b-1}$, derived as in theorem 2.2.1. Consider adding an additional stage of $b^{n-1}$ $(b \times b)$ switches denoted by $S^*$ in fig. 2.4.1. Suppose that stage $S_0$ of network $A$ has control index $c_0$. Then stage $S^*$ is connected to network $A$ as follows. The $b$ network output links of network $A$ whose labels differ only in bit $d_{c_0}$, are connected to the same $(b \times b)$ switch in stage $S^*$. The control index at stage $S^*$ is set to $c_0$. Stage $S_0$ of this augmented network, $A'$, has no control index, and packets arriving at input links of this stage are equiprobably routed to the $b$ output links of the switch they are incident on. The network output links of $A'$ are labelled using lemma 2.2.3 (i.e. a network output link label is obtained by an appropriate concatenation of switch output links on a path from any network output link to that network output link). Alternatively, a method similar to (v) in theorem 2.3.1 can be used to label network output links. Notice that this construction procedure is very similar to the construction of $(b^{n+1} \times b^{n+1})$ bidirectional delta networks from $(b^n \times b^n)$ bidirectional delta networks as specified in algorithm 2.3.1.

We observe from fig. 2.4.1 that stage $S_0$ provides a connection from each network input link to an input link of each of $A^0$, $\ldots$, $A^{b-1}$. Since these sub networks are themselves delta networks, there is a unique path from each of their input links to each of their output links. Thus, from the construction of stage $S^*$, it immediately follows that there are exactly $b$ paths from each network input link of $A'$ to
each of its network output links. Further, these paths are disjoint except for common switches in stages $S_0$ and $S^\ast$. It also follows from the construction procedure that the normal digit controlled routing can still be used (with equiprobable routing in stage $S_0$). Since only one additional stage is added in this construction, an $(N \times N)$ augmented delta network has an $O(N \log N)$ gate complexity.

If network $A$ in the above discussion is a bidirectional delta network then the above construction procedure produces a bidirectional augmented delta network. That is, if packets are equiprobably routed in the reverse direction at stage $S^\ast$ (i.e., if no reverse control index is associated with stage $S^\ast$), then there are $b$ disjoint paths in both directions and digit controlled routing can be used in both directions.

An example of using this procedure to construct a $(2^3 \times 2^3)$ augmented bidirectional delta network is shown in fig. 2.4.2. In this figure, the bidirectional delta network of figure 2.3.1 is augmented. Also shown in the figure are two paths that are disjoint (except for the network input and output links).

Now suppose that a switch is able to detect a faulty link. Also suppose that there is additional hardware to pass this information on to adjacent switches. Then, if a link becomes faulty, this information can be propagated to all switches in stage $S_0$ that have a path to that link. These switches can then route all forward packet flow (from network input to output links) along the alternate paths in the network.
Figure 2.4.2
An augmented bidirectional delta network.
Similarly, the information can be propagated to switches in stage S*, and all reverse packet flow (from network output to input links) can be routed along alternate paths. An example of this is shown in fig. 2.4.2. Here, the faulty link is indicated by a cross and links along which this information is propagated are indicated by bold lines. Thus a packet directed from network input link 011 to network output link 010, will be routed along the upper dashed path which does not pass through a marked link. It is easy to see that the worst case single link fault can make one of the sub networks $A^0$, $\ldots$, $A^{b-1}$, totally inoperative in one direction. Also, a $(b^n \times b^n)$ augmented delta network operating in this mode can sustain up to b link failures and still have at least one path from any network input link to any network output link. Of course, this results in degraded performance (as we will see in later chapters).

In this section we have informally discussed the construction and properties of augmented delta networks. It is easy to formalize the construction and prove the properties of these networks in much the same way as in the preceding sections.

2.5. Pruned delta networks

We have considered delta networks with the same number of network input and output links. In this section we consider pruning delta networks to obtain networks with differing numbers of network input and output links while retaining the digit controlled property of these
networks.

Our main interest is in the case in which \((b \times b), (b \times 1)\) and \((1 \times b)\) basic switches are assumed to be available and in which all basic switches in the same stage are assumed to be of the same type. Further, all the output links of switches are required to be connected (i.e., no floating links are allowed). These networks will be called regular pruned delta networks. The \((b \times 1)\) basic switches are referred to as arbitration switches (since they arbitrate in choosing one of several input links that are to be connected to the output link). A stage composed of arbitration switches is called an arbitration stage. The \((1 \times b)\) switches are called distribution switches and a stage with switches of this type is called a distribution stage. Stages composed of \((b \times b)\) switches are called square stages. \((b^m \times b^k)\) pruned delta networks of this type, with \(m < k\), are referred to as distribution networks while, networks with \(m < k\) are called arbitration networks. This terminology is similar to that in [Dennis75a].

Theorem 2.2.1 can be used to prune a \((b^n \times b^n)\) delta network to obtain an arbitration network. Fig. 2.5.1(a) shows the construction of an arbitration stage. Only one of the sub-delta networks identified in theorem 2.2.1 is retained. Suppose that stage \(S_k, 0 \leq k < n\), is chosen as an arbitration stage and suppose that \(c_k\) is the control index at that stage. Since an arbitration switch has a single output link, the switches in stage \(S_k\) of the pruned network make no routing decision. This implies that if the label of a network output link is \(d_{n-1} \ldots d_0\)
(a) Constructing an arbitration stage.

(b) Constructing a distribution stage

Figure 2.5.1 Constructing regular pruned delta networks.
then digit $d_k$ of this label can be deleted and the control indices
$\tilde{c}_k$
greater than $c_k$ decremented by one. At an arbitration stage the ratio
of the number of input to output links is $b$. Thus $(n-m)$ arbitration
stages are required. Any $(n-m)$ stages from $S_0, \ldots, S_{n-1}$ can be
chosen as arbitration stages to obtain different pruned delta networks.
There are $\binom{n}{m}$ different choices of arbitration stages leading to dif-
f erent networks with differing numbers of switches. As an example, the
$\binom{3}{2}$ choices of arbitration stages in pruning a $(2^3 \times 2^3)$ delta network
to obtain $(2^3 \times 2^2)$ delta networks are shown in fig. 2.5.2.

The type of networks that result from extreme choices of the arbi-
tration stages are shown in fig. 2.5.3. In fig. 2.5.3(a) the arbitration
stages are the last $(n-m)$ stages of the network. For this case,
the resulting network has $(b^{n-1} \cdot m)$ square switches and
\[
\sum_{k=m-1}^{n-1} b^k = \frac{b^n - b^{m-1}}{b-1}
\]
 arbitration switches. The other extreme case, shown in fig. 2.5.3(b), has the arbitration stages as the first $(n-m)$
stages of the network. This results in the same number of arbitration
 switches but, $(b^{m-1} \cdot m)$ square switches are required. All the other
choices of the arbitration stages lead to networks with the same number
of arbitration switches but with the number of square switches varying
between the above extremes.

This discussion leads us to conclude that the minimum number of
switches results when the first $(n-m)$ stages of the network are chosen
Notation for stage type:
(A) denotes an arbitration stage.
(S) denotes a square stage.

Figure 2.5.2 The 3 choices of arbitration stages in pruning a
$\left(2^3 \times 2^3\right)$ delta network to obtain $(2^3 \times 2^2)$ pruned delta networks.
Figure 2.5.3  Extreme cases for \( b^n \times b^m \), \( n > m \), pruned delta networks.
to be arbitration stages as in fig. 2.5.3(b). However, to obtain the same throughput from this network as from the other extreme case (fig. 2.5.3(a)), the square switches need to operate faster. This aspect will be discussed in detail in chapter 5. It will be shown that under certain (reasonable) assumptions the maximum throughput is not obtained at either of these extreme cases but for some intermediate choice of arbitration stages.

In a manner analogous to the above, theorem 2.3.1 can be used to obtain a distribution network from a bidirectional delta network. The construction of a distribution stage is shown in fig. 2.5.1(b). Comments similar to those for arbitration networks apply to this case.

2.6. Chapter Summary

In this chapter we have presented the topology and some properties of delta networks. The main points can be summarized as follows.

(i) Delta networks are a class of multistage interconnection networks that are either topologically equivalent to or include several classes of networks that have been proposed in the literature. This implies that the performance studies of the following chapters apply to other networks as well.

(ii) Extreme cases of \((N \times N)\) delta networks are the crossbar switch, which has \(O(N^2)\) gates and \(O(N)\) ideal bandwidth, and the single bus, which requires \(O(N)\) gates and \(O(1)\) bandwidth.
The case of interest is that of delta networks composed of fixed size basic switches that have $O(N \log N)$ gates and $O(N)$ ideal bandwidth. We will see in the following chapters that in realistic enviroments such networks have a bandwidth comparable to the full crossbar switch.

(iii) The main idea behind the definition of delta networks is that of allowing for local control (also referred to as distributed control) in routing packets through the network.

(iv) Delta networks can be constructed recursively from smaller networks. This means that the networks are extensible.

(v) The basic switches in a delta network can be replaced by delta networks composed of smaller switches. The resulting network is also a delta network.

(vi) Delta networks allow for the local routing of packets in one direction (i.e., from network input to network output links). Bidirectional delta networks allow local packet routing in both directions.

(vii) All bidirectional delta networks are topologically equivalent. This implies that they have the same performance.
Delta and bidirectional delta networks have a unique path from each network input link to each network output link. They can be augmented to have several disjoint paths from network input to output links retaining the local routing capability and the $O(N \log N)$ gate complexity. This leads to better reliability.

Delta networks have the same number of network and input links. They can be pruned to obtain networks with differing numbers of input and output links. There are many ways of pruning a delta network to obtain regular pruned delta networks. We will look at the performance of the different pruned networks later in the thesis.
CHAPTER 3

Unbuffered Delta Networks

3.1. Introduction

In this chapter we examine the performance of unbuffered delta, augmented delta and pruned delta networks in a packet communication environment. In an unbuffered network, passing information between two links requires that a connection be made between these links. This is opposed to buffered networks in which packets can be stored within the network for later forwarding. However, we allow buffers at the inputs of unbuffered networks. Such buffers are considered to be external to the network. A model for unbuffered delta networks with crossbar type basic switches has been studied in [Patel79a]. We present a simpler analysis for this model and extend it to more realistic models. We show how network characteristics obtained for the simple model can be used to obtain approximate estimates of performance for other environments and models. The analysis is also generalized to augmented and pruned delta networks. We present simulation results for networks with (2 x 2) switches. These compare well with the approximate analytical results. Some simulation results for analytically intractable models are presented.
3.2. The model and performance measures

In this chapter, we make the following assumptions about the environment and the operation of the networks.

(i) Packets arriving at network input links contain both the data to be transferred and the label of the network output link to which data is to be passed.

(ii) Packets are routed through the network using a different digit of the destination output link label at each stage of the network (as discussed in chapter 2).

(iii) It is assumed that buffers at network output links are emptied instantaneously.

(iv) All input packets are assumed to be independently and equiprobably directed to each network output link.

(v) The network is assumed to operate in time intervals of length \( t_{\text{slot}} \). It is assumed that in each \( t_{\text{slot}} \) interval a packet is placed in an empty buffer at an input link with probability \( P_0 \).

(vi) In each \( t_{\text{slot}} \) interval an attempt is made to pass all input packets along paths, identified by digit controlled routing, from the network input link to the destination network output link (as described in chapter 2). If a conflict occurs, one of the conflicting packets is selected and passed while the
others are rejected.

Model (i) In this model we assume that one of the conflicting packets is equiprobably selected and that rejected packets are lost.

Model (ii) In these models rejected packets are resubmitted in the next t_slot interval. Equiprobable selection from conflicting packets and a priority selection are modelled.

Assumptions (i) and (ii) are related to the local routing of packets through the delta networks. For local routing it is necessary for packets to carry information of the destination address. The digit controlled property of delta networks is used in routing packets as discussed in chapter 2. Assumption (iii) implies that the network is slower than the destination modules. That is, it is assumed that the destination modules can consume any packets delivered to them faster than the network can produce them. The networks are assumed to operate synchronously as indicated in assumption (v). It is assumed that all packet transfers through the network occur in intervals of t_slot. The assumption in (vi) of a rejected packet being lost is made to facilitate analysis. In a realistic environment, a rejected packet would be resubmitted in the next t_slot time interval. An approximate analysis for this case is presented. The approximate estimates are compared with simulation results for networks with (2 x 2) switches in which rejected packets are resubmitted. The analysis assumes an equiprobable selection
from conflicting packets. We compare this with simulation results of a model in which priority is given to packets that have been in the system longer.

Performance measures

We use the following criteria to assess the performance of these networks.

(i) The average throughput, TP, is the average number of packets put out in unit time. Thus, if \( N_T \) is the number of packets put out by the network in time T then

\[
TP = \lim_{T \to \infty} \frac{N_T}{T},
\]

if this limit exists.

(ii) The normalized throughput, NTP, is the ratio of TP to the maximum possible average throughput. The maximum throughput, MAXTP, is an upper bound on the TP for the environment described, and is the TP of an ideal network in which there are no collisions.

(iii) In model (i), the probability of acceptance, \( p_a \), is the probability that a packet at an input link is accepted (i.e., is selected for passage and passed) in a t_slot interval.
(iv) The average time between which packets at an input link are accepted, $t_a$, is used as a performance criterion in model (i).

(v) The average turn around time, $TT$, is the average time interval between the time at which a packet is placed in a buffer at a network input buffer and the time at which it is placed in a network output buffer. Suppose that the packets that are put out of the network are numbered 1, 2, ..., and that the $i^{th}$ packet takes time $TT(i)$ in passing through the network, then with $N_T$ as defined above,

$$TT = \lim_{T \to \infty} \frac{1}{N_T} \sum_{i=1}^{N_T} TT(i)$$

if this limit exists.

(vi) The normalized turn around time, $NTT$, is the ratio of $TT$ to the minimum possible turn around time, which for this model is $t_{slot}$.

The thruput is an indicator of the rate at which packets stream through the network for a specific environment. We will see that the thruput increases monotonically with $P_0$. The thruput obtained with $P_0 = 1$ gives us an estimate of the network bandwidth or the maximum rate at which packets can stream through the network. For systems in which the packet input rate is independent of the delay of packets in passing
through the network, the bandwidth is an important performance criterion. An example of such a system is the data flow machine proposed in [Dennis77a] where it is claimed that there is sufficient parallelism in many applications that the turn around times of packets through the interconnection network is not a bottleneck. The NTP provides an indicator of the effect of conflicts in the network. The NTP for \( p_0 = 1 \) estimates the maximum utilization of the network. The TT indicates how long the network delays packets. The TT is of particular importance in closed systems in which the input rate is linked to the delay through the network.

3.3. \((b^n \times b^n)\) Delta Networks

We first show that all delta networks of the same size and with the same type of basic switches have the same performance.

**Lemma 3.3.1** Consider an unbuffered \((b^n \times b^n)\) delta network with the environment and assumptions of section 3.2, model (i) and consisting of \((b \times b)\) crossbar type basic switches. Let \( p_k \) denote the probability of a packet arriving at an input link of a switch in stage \( S_k \), \( 0 \leq k < n \), in a \( t_{\text{slot}} \) time interval. Let \( p_n \) denote the probability of a packet arriving at a network output link. Then,
(i) The arrival of a packet at an input link of a switch is independent of the arrival of a packet at the other input link of the same switch.

(ii) Each packet that arrives at a switch input link is independently and equiprobably directed to each output link of that switch.

(iii) \( p_k, 0 \leq k < n, \) is the same for all switch input links in the same stage, and \( p_n \) is the same for all network output links.

(iv) \( p_k, 0 \leq k \leq n, \) is the same for all \( (b^n \times b^n) \) delta networks composed of crossbar type basic switches.

**Proof** Statements (i) and (ii) follow directly from assumptions (iv), (v), and (vi) of section 3.2 and lemma 2.2.2. Statements (ii) and (iii) follow from an easy induction on \( k \), using assumptions (iv), (v), and (vi) of section 3.2 and lemma 2.2.2.

Lemma 3.3.1 is used to establish the following theorem.

**Theorem 3.3.1** All \( (b^n \times b^n) \) delta networks, with crossbar type basic switches and with the environment described in section 3.2, model (i), have the same TP, \( p_a \) and \( t_a \).

**Proof** With the definitions of TP, \( p_a \) and \( t_a \) in section 3.2, and the notation of lemma 3.3.1, \( TP = (p_n \times b^n / t_{slot}) \). \( p_a \) is the
conditional probability that a packet is passed, given that it exists at an input buffer, and is thus given by, \( p_a = \frac{p_n}{p_0} \). \( t_a \) is a function of \( p_a \). Thus, the theorem follows directly from lemma 3.3.1.

The preceding theorem shows that, for model (i), all delta networks have the same performance. The following theorem shows that all bidirectional delta networks have the same performance for model (ii).

**Theorem 3.3.2** All \((b^n \times b^n)\) bidirectional delta networks with the environment described in section 3.2, model (ii) and with crossbar type basic switches have the same TP, \( p_a \), \( t_a \), TT, NTP and NTT.

**Proof.** Follows from the topological equivalence of bidirectional delta networks (theorem 2.2.3) and the assumptions in section 3.2.

We have not been able to establish whether or not the entire class of delta networks has the same performance for model (ii). The approximate analysis that we present later, however, is for all delta networks while the simulation results for model (ii) are for bidirectional delta networks. The agreement between them suggests that there is no significant difference in performance between topologically different delta networks for model (ii).

We first analyze model (i) of section 3.2. Let \( p_k \), \( 0 \leq k < n \) be as defined in lemma 3.3.1. From this lemma it follows that the number of
packets that arrive at the input links of any switch \( m \) in stage \( k \) of the network in a \( t_{\text{slot}} \) interval has a binomial distribution. Thus the probability of \( j \) packet arrivals at switch \( m \) in a \( t_{\text{slot}} \) interval is 
\[ \binom{b}{j} p_k^j (1 - p_k)^{b-j} \]. From lemma 3.3.1, each packet arriving at an input of switch \( m \) is independently and equiprobably directed to each output link of switch \( m \). Thus

\[
p_{k+1} = \sum_{j=0}^{b} \binom{b}{j} p_k^j (1 - p_k)^{b-j} (1 - \left( \frac{b-1}{b} \right)^j)
\]

\[
= \sum_{j=0}^{b} \binom{b}{j} p_k^j (1 - p_k)^{b-j} - \sum_{j=0}^{b} \binom{b}{j} \left( \frac{b-1}{b} p_k \right)^j (1 - p_k)^{b-j}
\]

\[
= 1 - (\frac{b-1}{b} p_k + 1 - p_k)^b
\]

\[
p_{k+1} = 1 - (1 - \frac{p_k}{b})^b \quad (3.3.1)
\]

The TP is \((p_n \times b^n) / t_{\text{slot}}\). The maximum throughput, MAXTP, that can be obtained if there were no conflicts in the network is \((p_0 \times b^n) / t_{\text{slot}}\). NTP is the ratio of TP to MAXTP (i.e., TP normalized with respect to MAXTP) and is given by \((p_n / p_0)\). \( p_a \) is the conditional probability that a packet is passed given that it exists at an input link. Thus, \( TP = (p_0 \times p_a \times b^n) / t_{\text{slot}} \) which yields,
\[ p_a = p_a / p_0 = NTP. \]

t_a can be obtained as follows. Let \( q_a = 1 - p_a \). Then from the assumption of a rejected packet being lost we have,

\[
t_a = t_{\text{slot}} \cdot \sum_{k=1}^{\infty} k p_a (q_a)^{k-1} = t_{\text{slot}} \cdot p_a \cdot \frac{d}{dq_a} \sum_{k=1}^{\infty} (q_a)^k
\]

\[
t_a = t_{\text{slot}} \cdot p_a \cdot \frac{d}{dq_a} \left( \frac{1}{1 - q_a} - 1 \right) = \frac{t_{\text{slot}}}{p_a}. \quad (3.3.2)
\]

Fig. 3.3.1 shows the variation of \( t_a / t_{\text{slot}} \) and NTP with \( \log_2 N \), where \( N = b^n \) is the number of input and output links of the network. We make the following observations. First, note that NTP decreases monotonically with an increase in the size of the network. This is because there are more conflicts as the network size increases. However, the rate of decrease of NTP with increase in \( N \) falls off very fast. (Note that the plot is of NTP vs. \( \log_2 N \) and \( N \) doubles for each point shown.)

The \( t_a \) increase almost linearly with \( \log_2 N \). The performance is better for a larger \( b \). However, as discussed in chapter 2, for a constant \( N \), the number of gates is \( O \left( \frac{b}{\log b} \right) \), while the difference in performance is relatively small. Fig. 3.3.2 shows the variation of NTP and \( t_a \) with \( \log_2 b \). We observe that, for large \( b \), NTP and \( t_a \) are relatively constant for the same number of stages.
Parameter: basic switch size (b).

Figure 3.3.1  NTP and $T_a/t_{\text{slot}}$ versus $\log_2 N$ for model (i), with $p_0 = 1$, and using crossbar type basic switches.
Parameter: number of stages (n)

Figure 3.3.2 NTP and $T_a/t_{\text{slot}}$ versus $\log_2 b$ for model (i), with $p_0 = 1$, and using crossbar type basic switches.
These observations can be predicted from the following. From the limiting value, \( \lim_{n \to \infty} (1 + \frac{z}{m})^n = e^z \) [Abramowitz72a] we have

\[
\lim_{b \to \infty} p_{k+1} = 1 - e^{-p_k} . \quad (3.3.1)
\]

Thus, for large \( b \), \( p_k \) depends only on the number of stages. Consider the limit,

\[
\lim_{p_k \to 0} \left( \frac{1}{p_{k+1}} - \frac{1}{p_k} \right) = \lim_{p_k \to 0} \frac{p_k - (1 - \frac{p_k}{b})b}{p_k \cdot (1 - (1 - \frac{p_k}{b})b)} .
\]

Differentiating the numerator and denominator twice we get,

\[
\lim_{p_k \to 0} \left( \frac{1}{p_{k+1}} - \frac{1}{p_k} \right) = \frac{b - 1}{2b} .
\]

It is easily shown (by considering the difference \( p_{k+1} - p_k \) ) that the sequence \( p_0, p_1, \ldots \) is bounded and monotonically decreasing and hence converges, and that \( \lim_{k \to \infty} p_k = 0 \). Thus,

\[
\lim_{n \to \infty} (t_a \text{ for } (b^{n+1} \times b^{n+1}) \text{ networks} - t_a \text{ for } (b^n \times b^n) \text{ networks})
\]

\[
\lim_{n \to \infty} \left( \frac{1}{p_{n+1}} - \frac{1}{p_n} \right) = \lim_{p_n \to 0} \left( \frac{1}{p_{n+1}} - \frac{1}{p_n} \right) = \frac{b - 1}{2b} \quad (3.3.3)
\]

This accounts for the asymptotic linearity of \( t_a \) with \( \log_2 N \). Fig. 3.3.1 shows that this linearity holds approximately even for small values of \( n \).

A typical plot of the variation of NTP with \( p_0 \) is shown in fig. 3.3.3. The graph shown is for the basic switch size (b) of 2 and with the number of stages as parameter. As would be expected, NTP and \( p_a \) fall as the load on the network increase. The asymptotic value of NTP as \( p_0 \) approaches zero is obtained as follows.

\[
\lim_{p_k \to 0} \left( \frac{p_{k+1}}{p_k} \right) = \lim_{p_k \to 0} \frac{1 - (1 - \frac{p_k}{b})^b}{p_k} = 1
\]

This implies that \( \lim_{p_0 \to 0} \) NTP = 1. This is intuitive since at very low loads one would expect very few collisions in the network. From eq. (3.3.1), it follows that for large \( b \), \( p_n \) and hence NTP, \( p_a \) and \( t_a \) depend only on the number of stages \( (n) \) and \( p_0 \). This implies that characteristics similar to those described in fig. 3.3.3 are independent of \( b \) for large \( b \).
Parameter: number of stages \( n \).

Figure 3.3.3  NTP versus \( p_0 \) for model (i), with basic switch size (b) of 2, and using crossbar type basic switches.
We now show how these characteristics can be used to obtain approximate estimates for other environments. Consider a \((b^n \times b^n)\) delta network with a FIFO queue of maximum length \(m\) at each network input link. Assume Bernoulli arrivals in each \(t_{\text{slot}}\) interval. (i.e., the probability of one packet arrival at a network input link in a \(t_{\text{slot}}\) interval is \(p_i = 1 - q_i\), and the probability of more than one arrival in a \(t_{\text{slot}}\) interval is 0.) The approximation we make is to assume that the occurrence of packets at different inputs of the network and their destination links are independent. We discuss a graphical approach to solving this problem. First assume that \(p_a\) is known. Then the input queue can be modelled by the finite state ergodic Markov chain shown in fig. 3.3.4. In this figure, state \(n_j\), \(0 \leq j \leq m\), is the state with \(j\) packets at a queue. We are interested in a steady state solution. Let \(p_{n_j}\) denote the steady state probability that an input queue is in state \(n_j\) in a \(t_{\text{slot}}\) interval. This chain is easily analysed leading to the following.

\[
p_k = \frac{p_0}{q_i} \left( \frac{p_i q_a}{p_a q_i} \right)^k \quad \text{for} \quad 1 \leq k \leq m \quad \text{and} \quad \frac{1}{q_i} \left( \frac{p_i q_a}{p_a q_i} \right)^{m+1} - \frac{p_i q_a}{p_a q_i} = 1
\]

(3.3.4)
Figure 3.3.4 Markov chain for an input buffer of length m, with Bernoulli input arrivals and assumed $p_a$. 

\[ \begin{align*}
\text{Figure 3.3.4} & \quad \text{Markov chain for an input buffer of length } m, \text{ with Bernoulli input arrivals and assumed } p_a. \\
\end{align*} \]
The effective loading on the network is the probability that at least one packet exists at an input link and is \( p_0 = (1 - p_{n_0}) \). Then,

\[
TP = \frac{(p_a \cdot p_0 \cdot b^n)}{t_{\text{slot}}} \quad \text{where} \quad p_0 = (1 - p_{n_0}) \quad (3.3.5)
\]

Using MAXTP as TP in a conflict free ideal network,

\[
\text{MAXTP} = \frac{(p_i \cdot b^n)}{t_{\text{slot}}} \quad \text{and} \quad \text{NTP} = \frac{p_a \cdot p_0}{p_i} \quad (3.3.5')
\]

TT can now be estimated using Little's result [Kleinrock75a]. If we denote the average number of packets in an input buffer by \( L \), Little's result gives,

\[
TT = \frac{L}{TP / b^n} \quad \text{where} \quad L = \sum_{j=1}^{m} j \cdot p_{n_j}
\]

Substituting for \( p_{n_j} \) from eq. 3.3.4 gives,

\[
TT = \frac{t_{\text{slot}} \cdot p_{n_0} \cdot r (m^{r+1} - (m+1)r^m + 1)}{(1 - p_{n_0}) p_a \cdot q_a \cdot (1 - p_{n_0}) (r - 1)^2} \quad (3.3.6)
\]

where \( r = \frac{p_i \cdot q_a}{p_a \cdot q_i} \)
The special cases of \( m = 1 \) and \( m = \infty \) will be examined. For \( m = 1 \), the preceding equations yield,

\[
p_{n_0} = \frac{p_a q_1}{p_a q_1 + p_1} \quad \text{and} \quad p_0 = 1 - p_{n_0} = \frac{p_1}{p_1 + p_a q_1}.
\]  \( (3.3.7) \)

TP and NTP are given by eq. 3.3.5 while TT simplifies to \( \frac{1}{p_a} \). Notice that as yet \( p_a \) is unknown. We use eq. 3.3.7 to obtain a characteristic of the input queues in the Bernoulli environment. Fig. 3.3.5 shows a graph of \( p_a \) vs. \( p_0 \) with \( p_1 \) as parameter. Superimposed are characteristics of some \( (2^n \times 2^n) \) delta networks. The intersection of the characteristics specifies the operating point and provides us with values of \( p_a \) and \( p_0 \). These are put into the equations for TP, NTP and TT. The resulting NTP and TT characteristics are shown in figs. 3.3.6 and 7, where they are compared with the results of simulation experiments.

For \( m = \infty \), the expressions for \( p_{n_0} \), TP, and TT converge only if

\[
r = \frac{p_1 q_a}{q_1 p_a} < 1.
\]

This implies that \( p_1 < p_a \) or the packet input rate to an input queue must be less than the rate at which packets are removed from the queue by the network. (Obviously, if this is not satisfied, infinite queues result.) If this condition is satisfied, the preceding equations yield

\[
p_{n_0} = \frac{p_a - p_1}{p_a}, \quad \text{NTP} = 1, \quad \text{TT} = \frac{t_{\text{slot}} (1 - p_1)}{p_a - p_1}, \quad \text{for} \ p_1 < p_a.
\]
Parameters: number of stages \( n \), and \( p_1 \).

\[\begin{array}{c}
\text{(p)} \\
1.0 \quad 0.1 \quad 0.2 \quad 0.3 \quad 0.4 \quad 0.5 \quad 0.6 \quad 0.7 \quad 0.8 \quad 0.9 \quad 1.0 \\
\end{array}\]

\[\begin{array}{c}
\text{(n)} \\
1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 10 \quad 12 \\
\end{array}\]

\[\begin{array}{c}
P_a \\
0.0 \quad 0.2 \quad 0.4 \quad 0.6 \quad 0.8 \quad 1.0 \\
\end{array}\]

\[\begin{array}{c}
P_0 \\
0.0 \quad 0.2 \quad 0.4 \quad 0.6 \quad 0.8 \quad 1.0 \\
\end{array}\]

Figure 3.3.5 \( p_a \) versus \( p_0 \) for a single buffer at network input links, basic switch size \( b \) of 2, Bernoulli input arrivals, and using crossbar type basic switches.
Parameter: number of stages \( n \).

- \( \bullet \) -- infinite buffers (analysis).
- \( \square \) -- single buffer (analysis).
- \( \times \) -- single buffer (simulation).

**Figure 3.3.6** NTP versus \( p_i \) for basic switch size of 2, Bernoulli input arrivals, and using crossbar type basic switches.
Parameter: number of stages (n)
- -- infinite buffers (analysis).
- -- single buffer (analysis).
* -- single buffer (simulation).

Figure 3.3.7  TT/t_slot versus p_1 for a basic switch size of 2, Bernoulli input arrivals, and using crossbar type basic switches.
The NTP is unity for this range as expected since if it were not, then infinite queues would result. If \( p_1 \geq p_a \), infinite queues result and \( p_0 = 1 \). Thus, for this range we have,

\[
NTP = \frac{p_n}{p_1}, \text{ for } p_0 = 1, \quad \text{when } p_1 \geq p_a
\]

The NTP characteristic that results is shown in fig. 3.3.6. TT can be estimated by the same method as before. Fig. 3.3.8 shows the \( p_a \) vs. \( p_0 \) characteristic of the infinite queues with \( p_1 \) as parameter, superimposed on the network characteristics. The operating point at steady state is estimated from the intersection of the characteristics. The resulting TT is graphed in fig. 3.3.7.

We make the following observations from these graphs. First, the approximate analytical estimate differs from simulation estimates by less than 7% for both NTP and TT. The analysis is simple and quick and can be generalized to other models for packet arrivals. The simulations require a lot of computer time. However, they allow more flexibility in modelling network operation.

As the length of buffers at the network input is increased, NTP increases but there is a concomitant increase in TT. At low loads, (i.e., small \( p_1 \)) there is a significant increase in NTP but at high loads, the increase is marginal. For large buffer lengths, TT becomes
Parameters: number of stages \((n)\), and \(p_1\).

- \(\times\) -- network characteristic.
- \(\times\) -- input buffer characteristic.

**Figure 3.3.8** \(p_a\) versus \(p_0\) for infinite buffers at network input links, basic switch size \((b)\) of 2, Bernoulli input arrivals, and using crossbar type basic switches.
very large as the maximum throughput that the network can support is approached. Thus, using small buffer lengths at the input forces the source to "slow down", and prevents very long waiting times.

We now present some simulation results for different models of operation of \( (2^n \times 2^n) \) delta networks. Graphs of NTP and NTT versus \( p_0 \) for four different models are shown in figs. 3.3.9 and 10. Case (a) in these figures is model (i) described above in which packets rejected in a \( t_{\text{slot}} \) interval are assumed to be lost. The simulation results for this case are very close to the analytical results reported earlier. Case (b) is for the model in which packets rejected in a \( t_{\text{slot}} \) interval are resubmitted from network input links in the next \( t_{\text{slot}} \) interval. (This is the model for which we obtained approximate analytical estimates.) In case (c), rejected packets are again resubmitted as in (b). However, in a conflict, priority is given to packets that have been in the system for a longer time. Case (d) models the policy in which a request by a packet is retained at the switch input at which it is in conflict with another packet, and loses in the arbitration. Such a packet then holds all links through which it has already passed successfully until it has passed to an output link of the network. Thus, any packet that needs to pass through any of these links, loses in the conflict at that link in the network.

The simulation results (figs. 3.3.9 and 10) show that the performances for cases (b) and (c) are almost the same. Actual values indicate that, for small networks, case (c) performs marginally better than
Figure 3.3.9 Simulation results for \((2^n \times 2^n)\) unbuffered delta networks with different operation policies.
Figure 3.3.10 Simulation results for \((2^n \times 2^n)\) delta networks with different operation policies.
case (b) while, for larger networks, case (b) performs slightly better than case (c). Also, for small networks, case (d) has almost the same values of NTT and NTP as cases (b) and (c) while for large networks the performance is perceptibly worse than that for cases (b) and (c). We have seen how the performance in (a) can be used to estimate the performance for case (b). A possible explanation for these trends is that introducing priority in conflict arbitration leads to a correlation between the existence of packets at different links of the same switch and this produces more frequent conflicts and smaller NTP.

3.4. Pruned and Augmented Delta networks.

We now turn to the estimation of the performance of augmented and regular pruned delta networks, under the assumptions of section 3.2. For augmented delta networks, we assume that packets are routed as described in section 2.4.

**Lemma 3.4.1** Statements (i), (ii) and (iii) of lemma 3.3.1 also hold for augmented and regular pruned delta networks operating with the assumptions of model (i).

**Proof** Analogous to the proof of lemma 3.3.1.

Consider a \((b₁ \times b₀)\) basic switch \(B\) at stage \(k\) of such a network (i.e., a basic switch with \(b₁\) input links and \(b₀\) output links). Let \(p_k\) be defined as in lemma 3.3.1. Then, as in the derivation of eq. 3.3.1,
\[ p_{k+1} = \sum_{j=1}^{b_i} \binom{b_i}{j} p_k^j (1 - p_k)^{b_i - j} (1 - \frac{b-1}{b_o})^j \]

which simplifies to

\[ p_{k+1} = 1 - (1 - \frac{p_k}{b_o})^{b_i} \quad (3.4.1) \]

This equation can be used to estimate the performance of these networks for model (i). The technique used in the previous section can then be used to obtain approximate estimates for the performance of these networks for other environments.

Recall from section 2.4 that an arbitration network is a \((b^n \times b^m)\) regular pruned delta network with \(n > m\). It is composed of \((n-m)\) arbitration stages constructed from \((b \times 1)\) switches and \(m\) square stages of \((b \times b)\) switches. Similarly, a distribution network is a \((b^m \times b^n)\) regular pruned delta network with \(n > m\) and is composed of \((n-m)\) distribution stages with \((1 \times b)\) switches and \(m\) square stages of \((b \times b)\) switches. As discussed in section 2.4, there are \(\binom{n}{m}\) different choices of arbitration/distribution stages leading to networks with differing numbers of switches. The following theorem shows that for the same choice of arbitration/distribution stages, all arbitration/distribution networks of the same size have the same performance.

**Theorem 3.4.1** Consider any two \((b^n \times b^m)\) arbitration
(distribution) networks $A$ and $B$ obtained by pruning any two $(b^a \times b^b)$ delta networks with the same choice of arbitration (distribution) stages (i.e., stage $S_k$, $0 \leq k < n$, in both $A$ and $B$ are of the same type).

Then, networks $A$ and $B$ have the same performance for the environment described in section 3.2, model (i).

**Proof** Follows from lemma 3.4.1 in a manner analogous to the proof of theorem 3.3.1.

The preceding theorem shows that the same choice of type of stages leads to the same performance. However, the performance for different choices of type of stages leads to differing performance. Eq. 3.4.1 can be used to obtain $p_n$. For a $(b^a \times b^b)$ arbitration/distribution network,

$$TP = \frac{p_n \times b^m}{t_{slot}}.$$  

MAXTP is estimated as $\min\left(\frac{p_0 \times b^n}{t_{slot}}, \frac{b^m}{t_{slot}}\right)$. The first term in this expression is for an ideal network with no conflicts, while the second term is the throughput obtained if the output links have a packet in each $t_{slot}$ interval. As in section 3.3.1,

$$TP = \frac{p_0 \times p_a \times b^n}{t_{slot}}$$

which yields $p_a = \frac{p_n \times b^m}{p_0 \times b^n}$. As before, $t_a = \frac{t_{slot}}{p_a}$.

Using these equations leads to the result that the extreme cases for the choice of arbitration/distribution stages (described in section 2.4) lead to extremes in performance. The performance increases monotonically with increase in the number of switches in the network. Thus,
the network with the best performance also requires the most switches.

For \((b^n \times b^m)\) arbitration networks, the best performance (i.e., maximum TP, NTP, \(p_a\) and minimum \(t_a\)) is obtained for the network with stages \(S_0 \ldots S_{n-1}\) as square stages and the rest as arbitration stages. The worst performance is for the arbitration network with stages \(S_0 \ldots S_{n-m-1}\) as arbitration stages and the rest as square stages. The other choices lead to intermediate performance. A typical case is for \(b = 2, n = 8\) and \(p_0 = 1.0\) and results for this case are shown in fig. 3.4.1. At extreme values of \(m\) there is a single choice for the types of stages. For intermediate values of \(m\) between the extremes, there is a considerable difference between best and worst NTP. However, there is a corresponding difference in the number of switches in the network as shown in fig. 3.4.1. For \(p_0 \cdot b^n \gg b^m\), \(p_a\) is very small and \(t_a\) is correspondingly very large. Thus networks with \(b^n \gg b^m\) do not have a built-in mechanism to prevent overloading. To obtain a reasonable \(p_a\) the input rate needs to be regulated. We will see in a later chapter how buffering within the network can alleviate the situation.

For \((b^n \times b^n)\) distribution networks, the best performance is exhibited by the network with stages \(S_0 \ldots S_{n-m-1}\) as distribution stages and the rest as square stages. The worst performance is for the network with the stages \(S_0 \ldots S_{m-1}\) stages as square stages and the rest as distribution stages. Results for a typical case are shown in
Figure 3.4.1 NTP, $p_a$, and number of switches versus the number of output links, for arbitration networks ($b = 2, n = 8, p_0 = 1$).
fig. 3.4.2. For this case, \( p_a = \text{NTP} \) and is quite acceptable. However, for \( b^n >> b^m \), \( p_n \) is very small and this indicates a poor utilization of the latter stages of the network. We will see later that for buffered networks it is feasible to 'match' the stages and prevent underutilization.

Next, consider augmented delta networks with normal (fault-free) operation, and with the assumptions of model (i). Lemma 3.4.1 and eq. 3.4.1 together imply that a \( (b^n \times b^n) \) augmented delta network that has \( (n + k) \) stages has the same \( p_{n+k} \) as a \( (b^{n+k} \times b^{n+k}) \) delta network. Since \( \text{NTP} \), \( p_n \) and \( t_a \) all depend only on \( p_{n+k} \) and \( p_0 \), a similar statement is also true for these criteria. We observed in section 3.3 that the rate of change in NTP with \( n \) decreases as \( n \) increases. Thus, for small \( k \), there is little deterioration in the normal (fault-free) performance of augmented delta networks as compared to the same size delta network. Fig. 3.4.4 shows the normal performance of some \( (2^n \times 2^n) \) augmented delta networks with \( n+1 \) stages.

We now look at the worst case performance of a \( (b^n \times b^n) \) augmented delta network with \( (n+1) \) stages, operating with a single faulty switch. This is easily generalized to \( (n+k) \) stages for \( k>1 \). Fig. 2.4.1 is repeated in fig. 3.4.3 for convenience. As discussed in section 2.4, the worst case internal switch fault results in one of the \( A^k \) networks becoming inoperative. The resulting network effectively has \( (b \times b-1) \) switches in stage \( S^0 \), \( (b-1 \times b) \) switches in stage \( S_{n-1} \) and \( (b \times b) \).
Figure 3.4.2  NTP, $p_a$, $p_n$, and number of switches versus the number of input links, for distribution networks ($b = 2$, $n = 8$, $p_0 = 1$).
Figure 3.4.3 Worst case single fault at an interior switch of an augmented delta network.
switches at other stages. $p_n$ can now be calculated using eq. 3.4.1. Results for some typical case are shown in fig. 3.4.4. For small $p_0$ the performance for the worst case fault is quite good. For large $p_0$ NTP for the worst case fault is a little more than one half NTP for normal operation.

3.5. Chapter Summary

In this chapter we have studied the performance of unbuffered delta networks with crossbar type basic switches. The main results of this chapter can be summarized as follows.

(i) All delta networks have the same performance for model (i) (i.e., when rejected packets are lost).

(ii) All bidirectional delta networks have the same performance for model (ii) (i.e., when rejected packets are resubmitted). The approximate analysis and simulation results suggest that there is no significant difference in the performance of any two delta networks for model (ii).

(iii) A simple model for unbuffered delta networks (model (i)) can be used to obtain characteristics of these networks that can then be used to obtain fairly good estimates of performance for other models and environments.
Parameter: number of stages (n)

Figure 3.4.4  NTP versus $p_0$ for augmented delta networks, model (1), $b = 2$, using crossbar type basic switches, under normal operation and for a worst case single fault.
(iv) For large basic switches, NTP and NTT depend only on the number of stages in the network.

(v) For fixed size basic switches, the normalized bandwidth of delta networks decreases monotonically with an increase in the size of the networks. However, the rate of decrease of the bandwidth with increase in network size falls off rapidly.

(vi) NTP and NTT for network operation policies in which priority is given to packets that have been in the network for a longer time, or where packets are resubmitted at the point of reject, is worse than that for the policy in which an equiprobable choice from conflicting packets is made.

(vii) For regular pruned delta networks, extremes in performance are obtained for extreme choices of arbitration or distribution stages, with better performance being obtained for networks with more switches.

(viii) The performance of augmented bidirectional delta networks with normal (fault free) operation is a little worse than that of the same size delta networks. The NTP for a worst case single fault is a little better than half that for normal operation.
CHAPTER 4

Buffered Delta Networks

4.1. Introduction

This chapter is a study of the performance of buffered delta networks in a packet communication environment. Unbuffered delta networks allow parallelism in the transfer of packets within a stage. Introducing buffers between the stages leads to a pipelining effect between stages. Thus, with buffering, there is parallelism in packet transfer both within a stage and in different stages. For an $n$-stage pipeline, an upper bound on the speedup due to pipelining is $n$. However, buffering in delta networks can lead to a better than $n$-fold improvement in the network bandwidth. This is because of a concomitant decrease in the collisions within the network.

The emphasis in this chapter is on a "maximum load" environment in which packets are placed in buffers at the input links of the network as soon as a buffer is available. This environment is of interest for several reasons. First, it leads to more congestion in the network than for any other environment. As a result the turn around time of packets passing through the network is longer for this environment than for any other. Further, the maximum network throughput is also obtained for this environment. Thus, this case provides information about the network
bandwidth and the concomitant worst case turn around times of packets passing through the networks. The performance of networks for this case also indicates whether further control of the networks is necessary. If the performance at a maximum load is unacceptable while a good performance is obtained at lower loads it suggests that the networks should be controlled to prevent excessive loading. Our study in this chapter indicates that for one or two buffers between stages the turn around times of packets in passing through the networks at a maximum load is quite acceptable while increasing the buffer length between stages leads to very large turn around times. This indicates that if a large number of buffers is used, the rate at which packets are fed to the network should be controlled.

We present the results of both approximate analytical and simulation estimates of the performance of delta networks in this environment. A performance estimation of these networks by simulation allows great flexibility in allowing for arbitrary arrival processes, switch operation policies, variation in model parameters, interconnection and loading patterns etc. However, the simulation of large networks requires large amounts of computer time and memory. Our resources restricted the size of the delta networks studied by simulation to a maximum of 256 network input and output links. An accurate analysis of the performance of these networks is very difficult, if not impossible. This is because of the extremely large number of distinguishable states in accurate models of these networks. In this chapter we present several approximate estimates of network performance. The approximations are substantiated
by simulation experiments for network sizes that can be simulated.

One of the aspects we study is the choice of the operation of a basic switch of the delta network. We observe that the network performance is sensitive to the switch operation policy. The choice of the mode of switch operation is influenced by values of certain parameters in a particular implementation.

The analysis is extended to pruned delta networks with buffers between stages. It is used to obtain regular pruned delta networks with the maximum throughput. The analysis is particularly important for this case since there are a large number ways of pruning delta networks and it is not feasible to simulate every choice, while the approximate analysis permits this. Some non intuitive optimal networks are derived.

4.2. The environment and performance criteria

The assumptions about the environment are the same as in chapter 3. Specifically, we make the assumptions (i), (ii), (iii) and (iv) of section 3.2. In addition we assume the following.

(v) If not otherwise stated, we consider an environment of "maximum loading". For this case, we assume the existence of a single buffer at a network input link that is filled by an input packet whenever it is emptied by the network.
The minimum delay at a basic switch is modelled as consisting of two time intervals. Time $t_{\text{select}}$ to select a switch output link and time $t_{\text{pass}}$ to pass the data to the selected output link. The minimum delay, $\text{min}_d$, of data through a $(b^n \times b^n)$ delta network is thus $n \times (t_{\text{select}} + t_{\text{pass}})$.

There are fixed length buffers between the stages of the network. These buffers are used as first in first out (FIFO) queues.

Assumption (v) essentially means that the interconnection network is the bottle-neck and that the source modules produce packets at least as fast as the network can carry them. The components of the minimum delay that a packet encounters at a basic switch are identified in (vi). By the minimum delay at a basic switch we refer to the delay of a packet through a switch when no other packet interferes with its passage through that switch. The time for extracting the control digit at that basic switch and requesting a transfer to the appropriate output is included in time $t_{\text{select}}$. The time to pass the packet to an empty buffer at that output, to clear the input buffer, and to inform the predecessor switch of an available buffer can be included in time $t_{\text{pass}}$. Assumption (vii) prescribes that the FIFO policy is used in operating the queues between the stages of a buffered delta network. This assumption is trivially true for single buffers between stages.

The performance measures that we use are the throughput $TP$, the turn around time $TT$, and these measures normalized with respect to their
maximum possible value (NTP and NTT). The definitions are the same as in section 3.2.

4.3. \((2^n \times 2^n)\) Buffered Delta Networks

In this section we study the performance of \((2^n \times 2^n)\) delta networks made up \((2 \times 2)\) crossbar type basic switches. Most of the delta networks that have been proposed in the literature are composed of \((2 \times 2)\) basic switches. An architecture that uses a bidirectional delta network as the memory processor interconnection network has been proposed in [Lundstrom80a; Barnes80a]. Prototypes of buffered \((2 \times 2)\) switches are being built at the Massachusetts Institute of Technology [Dennis80a], to be used in building "routing networks" which are \((2^n \times 2^n)\) delta networks. Thus, it is possible that \((2 \times 2)\) basic switches may soon be available on a chip to be used as building blocks for larger networks.

Some work on the performance of these networks has been done. Jacobsen and Misunas [Jacobsen77a] have done an approximate analysis of "routing networks" (which are special cases of delta networks) for low loads assuming Poisson arrival processes and infinite length buffers between stages. Loose bounds on the performance of routing networks have been developed in [Boughton79a; Misunas75a] where some simulation studies are also reported. We will present some relatively good analytical estimates of the performance of these networks. We first consider a model in which a packet waits at a switch until a buffer, at an output
link to which it is to be passed, becomes available. This model is modified to produce better throughput at a relatively small increase in turn around time. For each of these models we present approximate analyses and simulation results.

4.3.1 Model (1)

We specify the operation of the network for each model using timed Petri nets [Ramachandani74a] (see appendix 1). Shown in fig. 4.3.1 is a timed Petri net model for the operation of a (2 x 2) basic crossbar switch with a single buffer between stages. The operation is essentially as follows. There is a single buffer between stages and this is modelled by the places labelled as input buffers in fig. 4.3.1. The crossbar type basic switch can handle a packet at each input queue simultaneously. The model assumes that it takes time t_select to determine the output link to which the packet is to be passed. This time interval is modelled by transitions with a firing time of t_select. If the selected output is in use (i.e., another packet is in the process of being passed to that output link) the packet waits its turn for the use of that output link. This is modelled by the places labelled as output request flags. When the selected output link becomes available, the packet waits until a buffer at the successor input queue is empty. This information is provided by the places labelled as buffer empty flags. When both these conditions have been met, the packet is delayed by another time interval t_pass which models transmission delay. The transitions that take time t_pass to fire model this process. After this
Figure 4.3.1 Timed Petri net model for the operation of a (2 x 2) basic switch in model (i) (shown in the initial state).
time the data is available in the successor buffer.

This model is easily extended to bigger switches and longer buffers as shown in fig. 4.3.2. A fixed maximum queue length of waiting packets is allowed between stages. These queues are modelled by the subnets labelled as input queues in fig. 4.3.2. It is assumed that it takes negligible time to advance a packet in a queue since this can be accomplished by manipulating a pointer rather than by physical movement of data. Alternatively, for analytical purposes, this time to manipulate a queue can be included in the $t_{\text{pass}}$. Similarly, the backward propagation delay, in informing a predecessor switch of an empty buffer, can be included in $t_{\text{pass}}$.

4.3.1.1. Analysis

The analysis we present is for three special cases of $t_{\text{select}}$ and $t_{\text{pass}}$. Denote $(t_{\text{select}} + t_{\text{pass}})$ by $t_{\text{delay}}$. The three cases that we consider are for $t_{\text{select}} = 0$, $t_{\text{pass}} = 0$, and $t_{\text{select}} = t_{\text{pass}} = (t_{\text{delay}}/2)$. In a later section we compare the performance obtained with these assumptions with that for arbitrary values of these parameters. We will see that these three cases provide a good indicator of the performance for other parameter values.

We assume that at time = 0 the system is in the state with no packet in any buffer. With the above assumptions and the assumed operation of these networks, as described in the preceding section, it immediately follows that packets are placed in buffers only at times
Figure 4.3.2 Timed Petri net model for the operation of a \((b_1 \times b_0)\) basic switch with \(k\) buffers between stages and for model (1) (shown in the initial state).
that are integer multiples of $t_{\text{step}}$ where $t_{\text{step}}$ either equals $t_{\text{delay}}$ or $(t_{\text{delay}}/2)$. Let $t_k = (k \times t_{\text{step}})$, for $k = 0, 1, 2 \ldots$. The time interval $(t_k, t_{k+1}]$ is denoted by $\tau_k$.

We first consider the case of a single buffer between the stages of the delta network. The system can be modeled by a discrete time, aperiodic, Markov chain with a finite state space. This is possible because all state changes occur only at times $t_k$, for $k = 0, 1 \ldots$. However, an exact Markov chain model has an enormously large number of states making it analytically intractable except for trivially small networks. We develop first and second order approximate models based on the properties of delta networks that have been developed in chapter 2. These approximate models are substantiated by comparison with simulation results.

**Binomial approximation**

This approximation assumes that the number of packets at any stage of the networks is binomially distributed. Lemma 2.2.2 (iv,v) states that the same number of paths from network input links to network output links pass through each switch in a delta network. This fact and assumption (v) of the preceding section suggest that the probability of a packet existing at a buffer is the same for any buffer in a stage at the same instant of time. The binomial approximation assumes in addition that the events of packets existing at different buffers in the same stage are independent. We will see that this approximation gives
surprisingly good results when there are few stages in the network. Further, this approximate analysis can be generalized to \((b^n \times b^n)\) delta networks composed of \((b \times b)\) switches.

**Case 1**

First consider the case of \(t_{\text{select}} = 0\), and \(t_{\text{pass}} = t_{\text{step}} = t_{\text{delay}}\). In the following, if not otherwise stated, let \(k = 0, 1, \ldots, 0 \leq j < n\). Let

\[
p_j^k = 1 - q_j^k = \text{probability \{ a packet exists in a buffer at an input link of stage } j \text{ at time } t_k \}.
\]

The boundary conditions are \(p_0^k = 1\) and \(p_n^k = 0\), for \(k = 0, 1, \ldots\). Note that stage \(n\) does not exist in a \((2^n \times 2^n)\) delta network but \(p_n^k\) represents the probability that a packet exist at a network output link at time \(t_k\). These boundary conditions follow from assumptions (iii) and (iv) of the preceding section.

Some of the equations in this section will use a constant \(b\) which equals 2. Using this symbolism will allow us to use the equations developed in analysing \((b^n \times b^n)\) networks for values of \(b\) other than 2.

Consider a packet \(\zeta\) at an input buffer of a switch in stage \(j\) at time \(t_k\). The probability that there are \(m\) other packets incident on the same switch, \(0 \leq m < b\), at time \(t_k\) is \((b-1)\) \(p_j^k m\) \(1 - p_j^k \)\((b-1-m)\). The probability that \(k\) of these \(m\) packets are directed to the same output link
that packet $a$ is directed to is $\binom{m}{k} \left( \frac{1}{b} \right)^k (1 - \frac{1}{b})^{m-k}$, and $\binom{1}{k+1}$ is the probability that packet $a$ is selected for passage at this output. Thus,

$$p_j = 1 - q_j = \text{Probability} \{ \text{a packet at an input buffer of stage } j \text{ is passed in time interval } \tau_k \}$$

$$= q_{j+1}^k \left( \sum_{m=1}^{b-1} \binom{b-1}{m} \left( p_j \right)^m (1 - p_j)^{b-1-m} \left( \sum_{k=0}^{m} \binom{m}{k} \left( \frac{1}{b} \right)^k (1 - \frac{1}{b})^{m-k} \binom{1}{k+1} \right) \right).$$

In this equation, $q_{j+1}^k$ is the probability that the buffer at the output link to which packet $a$ is directed, is empty. The second summation is the probability that packet $a$ is selected for passage to that output. Some algebra yields,

$$p_j = q_{j+1}^k \left( \frac{1 - \left( 1 - \frac{p_j}{b} \right)^b}{p_j^k} \right).$$

Now consider an empty buffer $b$ at stage $j$, $0 < j < n$, which is on an output link of switch $M$ in stage $j-1$. From the assumed binomial distribution of packets, the probability that $m$ packets exist at the input links of switch $M$ is $\binom{b}{m} \left( p_{j-1} \right)^m (1 - p_{j-1})^{b-m}$. The probability that a packet is passed to buffer $b$ given that there are $m$ packets at the input links of $M$ is $1 - \left( \frac{b-1}{b} \right)^m$. Thus,
\[ p_j = 1 - q_j = \text{probability \{ an empty buffer at stage } j \text{ being filled in } \tau_k \} \]

\[ \approx \sum_{m=0}^{b} \binom{b}{m} \left(p_{j-1}^k\right)^m (1 - p_{j-1}^k)^{b-m} \left(1 - \frac{b-1}{b}\right)^m, \]

which simplifies to,

\[ p_j^k = 1 - \left(1 - \frac{p_{j-1}}{b}\right)^b. \quad (4.3.2) \]

The recurrence equation is,

\[ p_j^{k+1} = p_j^k q_j + q_j p_j^k, \quad (4.3.3) \]

for \(0 < j < n, k = 0, 1, \ldots\), with the boundary conditions,

\[ p_0^k = 1.0, \text{ and } p_n^k = 0.0. \quad (4.3.3)' \]

We are interested in the steady state solution to these equations. At steady state \( p_j^{k+1} = p_j^k \), producing a system of nonlinear equations.

The approach we take is the following. First, as discussed earlier, the system corresponds to a discrete time, aperiodic, Markov chain with a finite state space. Hence, for any fixed starting state, this system
reaches equilibrium with specific state probabilities. We want the approximate model to exhibit this property. Secondly, we are only interested in \((2^n \times 2^n)\) delta networks for a few values of \(n\). \((2^{16} \times 2^{16})\) delta networks are the maximum size that can be envisaged in the foreseeable future. Thus, it is easy to check on a computer whether the approximate model reaches a steady state.

The method we employ is to start at the network state with no packets in any buffer and to iterate using equation 4.3.3 until a steady state is reached. Recall that \(1 - \frac{w_j}{k}\) is the probability that a packet at an input buffer of stage \(j\) is passed in \(\tau_j\). Thus, \(\frac{p_j^k \tau_j^k}{p_j^k \tau_j^k}\) is the average number of packets that pass out of a buffer at an input link of stage \(j\) in time interval \(\tau_j\). The average number of packets that pass out of a buffer at an input link of stage \(j\) in time interval \(\tau_j\) is,

\[
NTP_j^k = p_j^k \tau_j^k
\]  

(4.3.4)

This steady state is detected by the condition that \(NTP_j^k\) is relatively constant for \(0 < j < n\) (i.e., all stages have the same average number of packets passing through them) and when this value is (relatively) constant over a large number of iterations (typically 100). The normalized throughput, \(NTP\), is estimated as the value of \(NTP_j^k\) at this steady state. Let \(p_j^k\) and \(\tilde{p}_j^k\) denote the values of \(p_j^k\) and \(p_j^k\) respectively at
steady state. Then using eq. 4.3.4 with \( j = n-1 \) yields,

\[
NTP \approx p_{n-1} \tilde{p}_{n-1}
\]  

(4.3.5)

The thruput, \( TP \), is estimated as \( (NTP \times 2^n / t_{\text{delay}}) \). An approximate estimate for the turn around time is obtained as follows. Recall that \( \tilde{p}_j \) is the steady state probability that a packet at an input buffer of stage \( j \) is passed in a \( \tau_k \) time interval. From the assumption of an equiprobable choice of conflicting packets, it follows that the average (steady state) delay that a packet encounters at stage \( j \) is,

\[
t_{\text{delay}} \cdot \sum_{k=1}^{\infty} k \tilde{p}_j (q_j)^{k-1} = \tilde{p}_j \cdot t_{\text{delay}} \cdot \frac{d}{dq_j} \sum_{k=1}^{\infty} (q_j)^k
\]

\[
= \frac{d}{dq_j} \left( \frac{1}{1 - q_j} \right) = \frac{t_{\text{delay}}}{p_j}.
\]

Thus, the turn around time, \( TT \), is estimated as,

\[
TT = t_{\text{delay}} \cdot \sum_{j=0}^{n-1} \frac{1}{p_j} \quad (4.3.6)
\]

and the normalized turn around time as

\[
NTT = \frac{1}{n} \cdot \sum_{j=0}^{n-1} \frac{1}{p_j}
\]  

(4.3.7)
The results are graphed in figs. 4.3.3-5, and will be discussed later.

Case 2

We now look at a binomial approximation for the case \( t_{\text{select}} = 1.0 \) and \( t_{\text{pass}} = 0.0 \). Consider advancing a packet from an input buffer of a stage to an input buffer of the succeeding stage \( \tau_k \) interval, \( k = 0, 1, \ldots \). The selection of the desired switch on a link in the \( t_{\text{select}} \) time interval occurs regardless of the availability of a buffer at that output link. For this case of \( t_{\text{pass}} = 0 \), a packet is advanced if it is selected at that output link and if a buffer is available at the end of the \( t_{\text{delay}} \) time interval. This in turn depends on whether a packet in that buffer was removed in the \( \tau_k \) interval and so on until an output stage is reached. Observe that, for this case, the last stage of the network delays all packets for the same \( t_{\text{delay}} \), even if two packets are directed to the same output link in the same time interval \( \tau_k \). This is because of assumption (iv) which states that packets at the network output links are removed instantaneously.

The analysis is facilitated by modelling the operation of the network in each \( \tau_k \) interval as consisting of two "logical" steps. In step 1, packets are advanced from a buffer to a succeeding stage. In step 2, packets are moved into a buffer from a preceding stage. Note that these "logical" steps are artificial and the separation has been made...
Figure 4.3.4
Binomial approximation: TP versus n.
Figure 4.3.5
Binomial approximation: NTT versus n.
facilitate computation. Note also that step 1 at stage $j$, $0 \leq j < n-1$, corresponds to step 2 at stage $(j+1)$. As discussed above, packets at the output of stage $(n-1)$ are always removed in step 1. Thus, given the probability that a packet exists at a link at time $t_k$, $k = 0, 1, \ldots$, one can compute the probability of a packet existing at a link at the input of stage $(n-1)$ after step 1 in time interval $\tau_k$. Similar computations at stage $j$, $0 \leq j < n-1$, can be made after they have been made at stage $(j+1)$. Similar comments apply to step 2. Thus, the probability that a buffer is occupied after each of these steps can be computed by starting at stage $(n-1)$ and proceeding back to stage 0. We use the following notation to formalize this procedure. In the following, if not otherwise stated, $k = 0, 1, \ldots, 0 \leq j < n$. Let,

$$p_j^k = \text{probability } \{ \text{a packet exists in a buffer at an input link of stage } j \text{ at time } t_k \} ,$$

$$p_{1j}^k = \text{probability } \{ \text{a packet exists in a buffer at an input link of stage } j \text{ in time interval } \tau_k \text{ after step 1} \} ,$$

and

$$p_{2j}^k = \text{probability } \{ \text{a packet exists in a buffer at an input link of stage } j \text{ in time interval } \tau_k \text{ after step 2} \} .$$

The boundary conditions are,

$$p_0^k = p_{10}^k = p_{20}^k = 1.0 ,$$

$$p_n^k = p_{1n}^k = p_{2n}^k = p_{n-1}^k = 0.0.$$
Let $\tilde{p}_j^k$, $q_j^k$, $p_j^k$, and $q_j^k$ be the same as defined earlier. Similar analysis to case (i) yields,

$$\tilde{p}_j^k = q_{j+1}^k \left( 1 - \frac{(1 - \frac{p_j^k}{b})^b}{p_j^k} \right), \text{ for } 0 \leq j < n-1, \quad (4.3.8)$$

and

$$\tilde{p}_{n-1}^k = 1.0. \quad (4.3.8)'$$

Note the similarity of this equation to eq. 4.3.1. The difference is that $q_{j+1}$ is replaced by $q_{j+1}^k$ and that packets at stage $n-1$ are always advanced. This means that the probability that an existing packet is advanced depends upon whether a packet exists at the buffer to which it is directed after step 1. $\tilde{p}_j^k$ is the same as in eq. 4.3.2. We now have,

$$p_{j}^{k+1} = p_{j}^{k} + (1 - p_{j}^{k}) \tilde{p}_{j}^{k}, \quad (4.3.9)$$

$$\text{for } 0 < j < n \text{ in both equations.}$$
Using the same technique as before, we start at the network state with no packets in any buffer, and iterate using eq. 4.3.9 until a steady state is reached. The steady state is detected using eq. 4.3.4, as for case 1. Using the same notation as before, the normalized throughput is estimated as,

$$NTP \approx P_{n-1} \quad (4.3.10)$$

Recall that for this case all packets at the input links of stage \( n-1 \) are advanced in the same \( t_k \) interval. The throughput is estimated as,

$$TP \approx \frac{2^n \cdot NTP}{t_{\text{delay}}} = \frac{2^n \cdot P_{n-1}}{t_{\text{delay}}} \quad (4.3.10)'$$

The absolute and normalized turn around times are estimated using equations 4.3.6, 7, and 8. The results are shown in figs. 4.3.3-5, and are discussed later.

**Case 3**

We now turn to a binomial estimate for the case of \( t_{\text{select}} = 0.5 \) and \( t_{\text{pass}} = 0.5 \). For this case \( t_{\text{step}} = (t_{\text{delay}}/2) \) (i.e., packets are placed in buffers only at times that are integer multiples of \( t_{\text{step}} \)). In the following, if not otherwise stated, \( k = 0, 1, \ldots \), and \( 0 \leq j < n \). We need to distinguish two states of a packet incident at an input buffer of a switch. Let,
\[ p_{ij}^k = \text{probability \{ a packet exists in a buffer at an input link of stage } j \text{ at time } t_k \text{ and is at the start of the } t_{\text{select}} \text{ phase } \}, \text{ and} \]
\[ p_{w}^k = \text{probability \{ a packet exists in a buffer at an input link of stage } j \text{ at time } t_k \text{ and is waiting for output passage } \}. \]

The latter state corresponds to a token in the places labelled as output request flags in figs. 4.3.1 and 2. Let \( p_j^k = p_{ij}^k + p_{w}^k \), and \( q_j^k = 1 - p_j^k \).

The boundary conditions are \( p_0^k = 1 \), and \( q_n^k = 1 \). Let,

\[ \sim p_j^k = \text{probability \{ a packet waiting for output passage at an input buffer of stage } j \text{ is passed in time interval } \tau_k \} \]

\[ \sim q_j^k = \frac{1 - (1 - \frac{p_w^k}{b})}{p_w^k} = 1 - q_j^k. \]  \hspace{1cm} 4.3.11

This equation is similar to eq. 4.3.1. Define,

\[ \sim p_j^k = \text{probability \{ an empty buffer at stage } j \text{ is filled in } \tau_k \} \]

\[ \sim 1 - (1 - \frac{p_{w}^k}{b})^b = 1 - q_j^k. \]  \hspace{1cm} 4.3.12
The derivation of this equation is similar to that of eq. 4.3.2. We now have,

\[
p_{w_j}^{k+1} = p_i^k + p_{w_j}^k \cdot q_j^k, \text{ for } 0 \leq j < n, \tag{4.3.13}
\]

\[
p_i^{k+1} = p_j^k \cdot q_j^k, \text{ for } 0 < j < n, \text{ and } p_i^0 = 1 - p_{w_0}^{k+1}. \tag{4.3.13'}
\]

The steady state is detected using the equation,

\[
NTP_j^k = p_{w_j}^k \cdot p_j^k, \tag{4.3.14}
\]

instead of eq. 4.3.4. Eliminating superscripts to denote steady state values, the normalized throughput is estimated as,

\[
NTP \approx 2 \cdot p_{w_{n-1}} \cdot p_{n-1}, \tag{4.3.15}
\]

which is similar to eq. 4.3.5 except for a factor of 2, which is introduced because \(t_{\text{step}} = t_{\text{delay}}/2\). The throughput is estimated as

\[
TP \approx \frac{2^n \cdot NTP}{t_{\text{delay}}}. \tag{4.3.15'}
\]

In estimating the TT, we observe that a packet is never delayed during the \(t_{\text{select}}\) phase. This leads to eq. 4.3.6 being modified to the
following equation.

\[ TT = \frac{t_{\text{delay}}}{2} \cdot \sum_{j=0}^{n-1} \left( 1 + \frac{1}{p_j} \right) \quad (4.3.16) \]

and \[ NTT = \frac{1}{2n} \cdot \sum_{j=0}^{n-1} \left( 1 + \frac{1}{p_j} \right) \quad (4.3.16)' \]

The derivation is similar to that of eq. 4.3.6. The results are graphed in figs. 4.3.3-5. The cases will be compared in detail later.

There are several advantages to the binomial approximation. First, as we will see, it provides an excellent estimate for networks with a few stages. For networks with many stages it indicates trends in performance. Thus, this approximation provides a reasonably accurate, quick and easily programmed estimate of performance. More importantly, it can be generalized to delta networks composed of larger crossbar type switches.

**Second order approximation**

The binomial approximation of the preceding section assumes that the existence of a packet in a buffer is independent of the existence of a packet in any other buffer at an input link of the same stage. In this section we allow the possibility that the events of packets existing at input buffers to the same switch are not independent. However,
we still assume that the incidence of packets at different switches in the same stage is independent. Lemma 2.2.2 (i) asserts that the set of network input links with a path to one of the links of a switch is disjoint from the corresponding set with a path to the other switch input link. We therefore make the approximation that the event that a packet is placed in a buffer at an input link of a switch is independent of the corresponding event at the other input link of the same switch, both at the same time $t_k$. We also assume that the event that a packet at the output of a switch is passed to a successor switch in time interval $t_k$ is independent of the passage of a packet at the other output of that switch.

Case 1

As before, we start by considering the case of $t_{\text{select}} = 0$ and $t_{\text{pass}} = t_{\text{step}} = t_{\text{delay}}$. With the above assumptions the distinguishable states of a $(2 \times 2)$ crossbar type basic switch with a single buffer between stages are as shown in fig. 4.3.6. These represent the states that a basic switch can assume. For $t_{\text{select}} = 0$, a packet at the input link of a switch, in one of these states, represents a token in a place labelled as an "output request flag" in fig. 4.3.1. Further, with the above assumptions, the two input (output) links of a switch are indistinguishable and this leads to a considerable reduction in the number of distinguishable states. Motivated by lemma 2.2.2 (v) we assume that all switches in the same stage have the same probability of being in each of
Notation: • denotes a packet and arrows show where packets are directed.

Figure 4.3.6 Internal states of a (2 x 2) switch.
these states at $t_k$, $k = 0, 1, \ldots$. We note that this lemma is true for all delta network. Thus, the following approximate analysis holds for the entire class of delta networks.

The method and notation that we use is similar to that used previously in the binomial approximation. In the following, if not otherwise stated, $k = 0, 1, \ldots$, $0 \leq j < n$, and $1 \leq i \leq 14$. Let,

$$p_{i,j}^k = \text{probability \{ switch in stage } j \text{ is in state } i \text{ at time } t_k \},$$

$$q_{i,j}^k = 1 - p_{i,j}^k,$$

$$p_{i,j}^k = \text{probability \{ a packet exists at an input link of a switch in stage } j \text{ at time } t_k \}$$

$$\approx \sum_{m=4}^{7} \frac{1}{2^m, j} + \sum_{m=8}^{14} p_{m,j}^k, \quad 4.3.17$$

$$p_{o,j}^k = \text{probability \{ a packet exists at an output link of a switch in stage } j \text{ at time } t_k \}$$

$$\approx \sum_{m=2, 5, 6, 10, 11, 12} \frac{1}{2^m, j} + \sum_{m=3, 7, 13, 14} p_{m,j}^k, \quad 4.3.18$$
\[ p_{\text{pass}}^k_j = \text{probability} \{ \text{a packet exists at an input link of a switch in stage } j \text{ at time } \tau_k \text{ and is passed to an output link of the same switch in } \tau_k \} \]

\[ = \sum_{m=4,6,8,11,12} \frac{1}{2^m} \text{p}^k_{m,j} + \sum_{m=9,11} \text{p}^k_{m,j}, \quad \text{and} \quad 4.3.19 \]

\[ p_{\text{fill}}^k_j = \text{probability} \{ \text{no packet exists at an output link of a switch in stage } j \text{ at time } \tau_k \text{ and it is filled by a packet in } \tau_k \} \]

\[ = \sum_{m=4,6,8,11,12} \frac{1}{2^m} \text{p}^k_{m,j} + \text{p}^k_{9,j}. \quad 4.3.20 \]

All these equations follow easily from the above assumptions and the definitions of the switch states shown in fig. 4.3.6. Define,

\[ \tilde{p}_j^k = \text{probability} \{ \text{a packet at a switch output link in stage } j \text{ is passed in } \tau_k \}, \quad \text{and} \quad \tilde{q}_j^k = 1 - \tilde{p}_j^k. \]

\[ \tilde{p}_j^k \text{ is merely a conditional probability that a packet is passed given that such a packet exists and is hence estimated by the following equation.} \]

\[ \tilde{p}_j^k \approx \frac{p_{\text{pass}}^k_{j+1}}{p_{\text{fill}}^k_{j+1}}, \quad \text{for } 0 \leq j < n-1, \quad 4.3.21 \]
with the boundary condition,

\[
\hat{p}_{n-1}^k = 1
\]  

(4.3.21)

Also define,

\[
\hat{p}_j^k = \text{probability \{ a packet from an input link at stage } j-1 \text{ is placed in a buffer at an input link of stage } j \text{ in } \tau_k \}, \quad \text{and} \quad \hat{q}_j^k = 1 - \hat{p}_j^k.
\]

\(\hat{p}_j^k\) is the conditional probability that a buffer is placed in a buffer given an empty buffer, and is approximated by,

\[
\hat{p}_j^k = \frac{p_{\text{fill}}^{j-1}}{1 - p_0^{j-1}} \quad , \text{for } 0 < j \leq n-1,
\]

(4.3.22)

with the boundary condition,

\[
\hat{p}_0^k = 1.0
\]

(4.3.22)'

The probabilities \(\hat{p}_j^k\) and \(\hat{q}_j^k\) can be used to compute state transition probabilities. An example is shown in fig. 4.3.7. In this example we consider a switch in stage \(j\) to be in state 7. The possible next states
Figure 4.3.7  An example of state transitions.
and the transition probabilities are shown. For example, the transition probability to state 10 is $0.5 \hat{p}_j \hat{q}_j$. The factor $\hat{p}_j \hat{q}_j$ is the probability that a specific output packet is advanced. The term $\hat{p}_j$ is the probability that another packet arrives at the empty input link while the 0.5 is the probability that this new packet is directed to the same output link as the existing packet. Thus, state transition tables can be constructed and these are shown in fig. 4.3.8. The state transition table for stage 0, denoted by $T_0^1$ is shown in fig. 4.3.8(a). The next state probabilities are estimated as,

$$ p_{i,0}^{k+1} = \sum_{m=1}^{14} p_{m,0}^k T_0^1(m,i) $$

for $1 \leq i \leq 14$, and $k = 0, 1, \ldots$.  \hspace{1cm} 4.3.23

Similarly, the state transition table, $T_j^2$, for stage $j$, $0 < j < n-1$, appears in fig. 4.3.8(b), and

$$ p_{i,j}^{k+1} = \sum_{m=1}^{14} p_{m,j}^k T_j^2(m,i) $$

for $1 \leq i \leq 14$, $k = 0, 1, \ldots$, and $0 < j < n-1$. (4.3.23)

The state transition table for stage $n-1$ is shown in fig. 4.3.8(c) and,
Transition matrix $T^1_0$

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>$\frac{1}{2}$</td>
<td>0</td>
<td>$\frac{1}{2}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\frac{1}{4}\bar{p}_j$</td>
<td>$\frac{1}{2}\bar{p}_j$</td>
<td>$\bar{q}_j$</td>
<td>$\frac{1}{4}\bar{p}_j$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\frac{1}{2}\bar{p}_j^2$</td>
<td>$\frac{1}{2}\bar{p}_j$</td>
<td>$\frac{1}{2}\bar{q}_j$</td>
<td>$\bar{p}_j\bar{q}_j$</td>
<td>$\frac{1}{2}\bar{p}_j\bar{q}_j$</td>
<td>$\frac{1}{2}\bar{q}_j^2$</td>
<td>$\frac{1}{2}\bar{q}_j^2$</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>$\frac{1}{2}$</td>
<td>0</td>
<td>$\frac{1}{2}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\frac{1}{2}\bar{p}_j$</td>
<td>$\frac{1}{2}\bar{q}_j$</td>
<td>0</td>
<td>0</td>
<td>$\frac{1}{2}\bar{q}_j$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\frac{1}{2}\bar{p}_j$</td>
<td>$\frac{1}{2}\bar{q}_j$</td>
<td>$\bar{q}_j$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\frac{1}{2}\bar{p}_j^2$</td>
<td>$\frac{1}{2}\bar{p}_j$</td>
<td>$\frac{1}{2}\bar{q}_j$</td>
<td>$\bar{p}_j\bar{q}_j$</td>
<td>$\frac{1}{2}\bar{p}_j\bar{q}_j$</td>
<td>$\frac{1}{2}\bar{q}_j^2$</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\bar{p}_j$</td>
<td>0</td>
<td>$\bar{q}_j$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\bar{p}_j$</td>
<td>$\bar{q}_j$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\bar{p}_j$</td>
<td>0</td>
<td>$\bar{q}_j$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\bar{p}_j^2$</td>
<td>0</td>
<td>$\bar{p}_j\bar{q}_j$</td>
<td>0</td>
<td>$\bar{p}_j\bar{q}_j$</td>
<td>$\bar{q}_j^2$</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\bar{p}_j^2$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\bar{q}_j^2$</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 4.3.8(a) State transition probabilities at stage 0 for $t_{\text{select}} = 0$. 
### Transition Matrix $T_j^2$

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$q_j^2$</td>
<td>0</td>
<td>0</td>
<td>$2\rho_j q_j$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\frac{1}{2}\rho_j^2$</td>
<td>$\frac{1}{2}\rho_j^2$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>$q_j^2 \rho_j$</td>
<td>$q_j^2 \rho_j$</td>
<td>0</td>
<td>$2\rho_j q_j$</td>
<td>$\rho_j \rho_j$</td>
<td>$\rho_j \rho_j$</td>
<td>0</td>
<td>$\frac{1}{2}\rho_j^2 \rho_j$</td>
<td>$\frac{1}{2}\rho_j^2 \rho_j$</td>
<td>$\frac{1}{2}\rho_j^2 \rho_j$</td>
<td>$\frac{1}{2}\rho_j^2 \rho_j$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>$q_j^2 \rho_j$</td>
<td>$2q_j \rho_j \rho_j$</td>
<td>$q_j^2 \rho_j$</td>
<td>$2q_j \rho_j \rho_j$</td>
<td>$2q_j \rho_j \rho_j$</td>
<td>$2q_j \rho_j \rho_j$</td>
<td>$2q_j \rho_j \rho_j$</td>
<td>$\frac{1}{2} \rho_j^2 \rho_j$</td>
<td>$\rho_j \rho_j$</td>
<td>$\rho_j \rho_j$</td>
<td>$\rho_j \rho_j$</td>
<td>$\rho_j \rho_j$</td>
<td>$\rho_j \rho_j$</td>
<td>$\frac{1}{2} \rho_j^2 \rho_j$</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>$q_j$</td>
<td>0</td>
<td>0</td>
<td>$\frac{1}{2} \rho_j$</td>
<td>$\frac{1}{2} \rho_j$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$q_j \rho_j$</td>
<td>$q_j \rho_j$</td>
<td>0</td>
<td>0</td>
<td>$\frac{1}{2} \rho_j \rho_j$</td>
<td>$\frac{1}{2} \rho_j \rho_j$</td>
<td>$\frac{1}{2} \rho_j \rho_j$</td>
<td>$\frac{1}{2} \rho_j \rho_j$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>$q_j \rho_j$</td>
<td>$q_j \rho_j$</td>
<td>0</td>
<td>$\frac{1}{2} \rho_j \rho_j$</td>
<td>$\frac{1}{2} \rho_j \rho_j$</td>
<td>$\rho_j \rho_j$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$q_j \rho_j$</td>
<td>$q_j \rho_j$</td>
<td>$q_j \rho_j$</td>
<td>$q_j \rho_j$</td>
<td>$\frac{1}{2} \rho_j \rho_j$</td>
<td>$\frac{1}{2} \rho_j \rho_j$</td>
<td>$\frac{1}{2} \rho_j \rho_j$</td>
<td>$\frac{1}{2} \rho_j \rho_j$</td>
<td>$\frac{1}{2} \rho_j \rho_j$</td>
<td>$\frac{1}{2} \rho_j \rho_j$</td>
<td>$\frac{1}{2} \rho_j \rho_j$</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\rho_j$</td>
<td>0</td>
<td>$\bar{q}_j$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\rho_j$</td>
<td>$\bar{q}_j$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\rho_j$</td>
<td>$\bar{q}_j$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\rho_j$</td>
<td>$\bar{q}_j$</td>
<td>0</td>
<td>$\rho_j \bar{q}_j$</td>
<td>0</td>
<td>$\rho_j \bar{q}_j$</td>
<td>$\bar{q}_j^2$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\rho_j^2$</td>
<td>0</td>
<td>$2\rho_j q_j$</td>
<td>0</td>
<td>0</td>
<td>$q_j^2$</td>
</tr>
</tbody>
</table>

Figure 4.3.8(b) State transition probabilities at stage $j$, $0 < j < n-1$, for $\epsilon_{\text{select}} = 0$. 

134
Transition matrix $T_{n-1}^3$

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$q_j^2$</td>
<td>0</td>
<td>0</td>
<td>$2q_j q_j$</td>
<td>0</td>
<td>0</td>
<td>$\frac{1}{2}q_j^2$</td>
<td>$\frac{1}{2}q_j^2$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>$q_j^2$</td>
<td>0</td>
<td>0</td>
<td>$2q_j q_j$</td>
<td>0</td>
<td>0</td>
<td>$\frac{1}{2}q_j^2$</td>
<td>$\frac{1}{2}q_j^2$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>$q_j^2$</td>
<td>0</td>
<td>0</td>
<td>$2q_j q_j$</td>
<td>0</td>
<td>0</td>
<td>$\frac{1}{2}q_j^2$</td>
<td>$\frac{1}{2}q_j^2$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>$q_j$</td>
<td>0</td>
<td>0</td>
<td>$\frac{1}{2}q_j$</td>
<td>$\frac{1}{2}q_j$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>$q_j$</td>
<td>0</td>
<td>0</td>
<td>$\frac{1}{2}q_j$</td>
<td>$\frac{1}{2}q_j$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>$q_j$</td>
<td>0</td>
<td>0</td>
<td>$\frac{1}{2}q_j$</td>
<td>$\frac{1}{2}q_j$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>$q_j$</td>
<td>0</td>
<td>0</td>
<td>$\frac{1}{2}q_j$</td>
<td>$\frac{1}{2}q_j$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 4.3.8(c) State transition probabilities at stage (n-1) for $t_{select} = 0$. 
\[ p_{i,n-1}^{k+1} = \sum_{m=1}^{14} p_{m,n-1}^{k} T_{n-1}^{2}(m,i) , \]

for \( 1 \leq i \leq 14 \), and \( k = 0, 1, \ldots \) \hspace{1cm} (4.3.23)''

We use the same technique as for the binomial approximation. We start at the network state with no packets in any buffer. This means that \( p_{1,j}^{0} = 1.0 \) for \( 0 \leq j < n \). Observe from the above definitions that

\[ \text{NTP}_{j}^{k} = (p_{0,j}^{k} \cdot p_{j}^{k}) \]

is the average number of packets that pass out of an output link of a switch in stage \( j \) in time interval \( \tau_{k} \). Eq. 4.3.23 is used to iterate until a steady state is reached. The steady state is detected, as before, by the condition that \( \text{NTP}_{j}^{k} \) is (relatively) constant for \( 0 \leq j < n \) (i.e. all stages have the same average number of packets passing through them), and when this value is (relatively) constant over a large number of iterations (typically 100). Eliminating superscripts to denote steady state values, the normalized thruput is estimated as,

\[ \text{NTP} \approx p_{0,n-1} \], and \hspace{1cm} (4.3.25)

\[ \text{TP} \approx \frac{2^{n} \cdot \text{NTP}}{t_{\text{delay}}} = \frac{2^{n} \cdot p_{1,n-1}}{t_{\text{delay}}} \] \hspace{1cm} (4.3.25)'
The TT is approximated in an analogous manner to that for the binomial approximation. Observe from the definition of $p_j^k$ and eq. 4.3.21 that $p_j^k$ is the probability that a packet at input buffer of stage $(j+1)$ is passed in interval $\tau_k$. Thus as in the derivation of eq. 4.3.6 the turn around time is estimated as,

$$TT \approx t_{\text{delay}} \cdot \sum_{j=0}^{n-1} \sum_{k=1}^{k-1} p_{j-1}^k (q_{j-1})^{k-1}$$

$$TT \approx t_{\text{delay}} \cdot \sum_{j=0}^{n-1} \frac{1}{p_{j-1}}$$

and $$NTT \approx \frac{1}{n} \cdot \sum_{j=0}^{n-1} \frac{1}{p_{j-1}} \quad (4.3.26)'$$

NTP is shown in fig. 4.3.9 and NTT is shown in fig. 4.3.10. Each is also compared with results from the binomial approximation.

**Case 2**

We now consider the case of $t_{\text{select}} = t_{\text{delay}} = t_{\text{step}}$ and $t_{\text{pass}} = 0$. The method used is similar to that for the binomial approximation. Recall that, for this case, the last stage in the network delays all packets for a time of $t_{\text{delay}}$ even if two packets are directed to the same output in time interval $\tau_k$. This is because of the
Figure 4.3.9: Second order and Binomial approximation: NTP versus n.
Figure 4.3.10  Second order and Binomial approximations: NTT versus n.
assumption that packets at the output of the network are removed instantaneously. Thus, for this case, the last stage of the network does not affect TP and causes a fixed delay. Thus it can be eliminated from the analysis. The probability of a packet at an input of stage \( j \) being advanced in \( \tau_k \) depends on whether packets at succeeding stages were advanced.

The analysis is facilitated by modelling the operation of the network in each \( \tau_k \) interval as consisting of three "logical" steps. In step 1, packet movement at the output of switches occurs. In step 2, packets move from input links of switches to output links of the corresponding switches. In step 3, packets move into buffers at switch input links from preceding stages. Note that these "logical" steps are artificial and the separation has been made to facilitate computation. Note also that from the definition of the states of a switch (fig. 4.3.6), step 1 at stage \( j \), \( 0 \leq j < n-1 \), corresponds to step 2 at stage \( (j+1) \), and step 3 at stage \( (j+1) \) corresponds to step 2 at stage \( j \).

Now, packets at the output of stage \( (n-2) \) are always removed in step 1. Thus, given the state probabilities of switches at \( t_k \), \( k = 0, 1 \ldots \), one can compute the state probabilities of switches in stage \( (n-2) \) after step 1 in time interval \( \tau_k \). Similar computations at stage \( j \), \( 0 \leq j < n-2 \), can be made after they have been made at stage \( (j+1) \). Similar comments apply to steps 2 and 3. Thus the state probabilities after each of these steps can be computed by starting at stage \( n-2 \) and proceeding back to stage 0.
The procedure can be formalized as follows. Let $p_{i,j}^k$, $p_{j}^-^k$ and $p_{j}^k$ be defined as for the previous case. Define,

\[ p_{i,j}^1 = \text{probability \{ switch in stage } j \text{ is in state } i \text{ in time interval } \tau_k \text{ after step 1 \}}, \]

\[ p_{i,j}^2 = \text{probability \{ switch in stage } j \text{ is in state } i \text{ in time interval } \tau_k \text{ after steps 1 and 2 \}}, \]

\[ p_{i,j}^3 = \text{probability \{ switch in stage } j \text{ is in state } i \text{ in time interval } \tau_k \text{ after steps 1, 2 and 3 \}}. \]

Define $p_{i,j}^k$, $p_{i,j}^k$, $p_{i,j}^k$ and $p_{i,j}^k$ to be the probabilities, after step 1, that correspond to $p_{i,j}^k$, $p_{i,j}^k$, $p_{i,j}^k$, and $p_{i,j}^k$ respectively. These probabilities can be computed using eqs. 4.3.17-20 with $p_{i,j}^k$ replaced by $p_{i,j}^k$. Then,

\[ p_{j}^-^k = \frac{p_{i,j}^{\text{pass}^k}}{p_{i,j}^{\text{pass}^k} + p_{i,j}^{\text{fill}^k}}, \text{ for } 0 \leq j < n-2, \]

4.3.27

with the boundary condition,

\[ p_{j}^-^n = p_{j}^-^{n-2} = 1 \]

(4.3.27)'
\[ p_j^k = \frac{p_{fill}^k}{1 - p_{lo}^k} \] for \( 0 < j < n-2 \), \( 4.3.28 \)

with the boundary condition,

\[ p_0^k = 1.0 \] \( (4.3.28)' \)

State transition tables for each of the steps can be derived as for the previous case and these are shown in fig. 4.3.11. The transition tables for steps 1, 2 and 3 are denoted by \( T_j^4 \), \( T_j^5 \) and \( T_j^6 \) respectively. The limits are \( 0 \leq i \leq 14 \), \( 0 \leq j < n-1 \) and \( k = 0, 1, \ldots \). The transition equations are,

\[ p_{1i,j}^k = \sum_{m=1}^{14} p_{mj}^k T_j^4(m,i) \] \( (4.3.29) \)

\[ p_{2i,j}^k = \sum_{m=1}^{14} p_{mj}^k T_j^5(m,i) \], and \( (4.3.29)' \)

\[ p_{3i,j}^{k+1} = p_{3i,j}^k = \sum_{m=1}^{14} p_{2mj}^k T_j^6(m,i) \] \( (4.3.29)'' \)

The average throughput is,

\[ NTP \approx p_{n-2}^0 \] and \( TP \approx \frac{2^n \cdot p_{n-2}^0}{t_{\text{delay}}} \) \( 4.3.30 \)
Transition matrix $T_j^4$

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>$\bar{p}_j$</td>
<td>$\bar{q}_j$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>$\bar{p}_j^2$</td>
<td>$2\bar{p}_j\bar{q}_j$</td>
<td>$\bar{q}_j^2$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\bar{p}_j$</td>
<td>$\bar{q}_j$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\bar{p}_j$</td>
<td>0</td>
<td>$\bar{q}_j$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\bar{p}_j^2$</td>
<td>$\bar{p}_j\bar{q}_j$</td>
<td>$\bar{p}_j\bar{q}_j$</td>
<td>$\bar{q}_j^2$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\bar{p}_j$</td>
<td>0</td>
<td>$\bar{q}_j$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\bar{p}_j$</td>
<td>0</td>
<td>$\bar{q}_j$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\bar{p}_j$</td>
<td>0</td>
<td>0</td>
<td>$\bar{q}_j$</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\bar{p}_j^2$</td>
<td>0</td>
<td>$\bar{p}_j\bar{q}_j$</td>
<td>0</td>
<td>$\bar{p}_j\bar{q}_j$</td>
<td>$\bar{q}_j^2$</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\bar{p}_j^2$</td>
<td>0</td>
<td>2$\bar{p}_j\bar{q}_j$</td>
<td>0</td>
<td>0</td>
<td>$\bar{q}_j^2$</td>
</tr>
</tbody>
</table>

Figure 4.3.11(a) State transition probabilities for step 1 with $t_{pass} = 0$. 
Transition matrix $T^5_{i,j}$

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 4.3.11(b) State transition probabilities for step 2 with $t_{pass} = 0$. 
Transition matrix \( T^6_j \)

\[
\begin{array}{cccccccccccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 \\
1 & q_j^2 & 0 & 0 & 2p_jq_j & 0 & 0 & 0 & \frac{1}{2}p_j^2 & \frac{1}{2}q_j^2 & 0 & 0 & 0 & 0 \\
2 & 0 & q_j^2 & 0 & 0 & p_jq_j & p_jq_j & 0 & 0 & 0 & \frac{1}{4}p_j^2 & \frac{1}{2}p_j & \frac{1}{4}p_j^2 & 0 & 0 \\
3 & 0 & 0 & q_j^2 & 0 & 0 & 0 & 2p_jq_j & 0 & 0 & 0 & 0 & \frac{1}{2}p_j^2 & \frac{1}{2}q_j^2 & 0 \\
4 & 0 & 0 & 0 & q_j & 0 & 0 & 0 & \frac{1}{2}p_j & \frac{1}{2}p_j & 0 & 0 & 0 & 0 & 0 & 0 \\
5 & 0 & 0 & 0 & 0 & q_j & 0 & 0 & 0 & 0 & \frac{1}{2}p_j & \frac{1}{2}p_j & 0 & 0 & 0 & 0 \\
6 & 0 & 0 & 0 & 0 & 0 & q_j & 0 & 0 & 0 & 0 & \frac{1}{2}p_j & \frac{1}{2}p_j & 0 & 0 & 0 \\
7 & 0 & 0 & 0 & 0 & 0 & 0 & q_j & 0 & 0 & 0 & 0 & \frac{1}{2}p_j & \frac{1}{2}p_j & 0 & 0 \\
8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
11 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
13 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\end{array}
\]

Figure 4.3.11(c) State transition probabilities for step 3 with 
\( t_{\text{pass}} = 0 \).
The TT and NTT are estimated using eq. 4.3.26. The resulting performance measures are graphed in figs. 4.3.9 and 10.

**Case 3**

We now turn to the case of \( t_{\text{select}} = t_{\text{pass}} = (t_{\text{delay}}/2) \). For this case \( t_{\text{step}} = (t_{\text{delay}}/2) \). (i.e., packets are placed in buffers only at times that are integer multiples of \( t_{\text{step}} \).) The distinguishable states of a \((2 \times 2)\) crossbar switch for this case are shown in fig. 4.3.12. The increase in the number of states from those for the previous cases is because we need to distinguish between a packet corresponding to a token at an input buffer (fig. 4.3.1) and at an output request flag. The notation is explained in fig. 4.3.12. We note in particular that a packet at an output link of a switch in this figure corresponds to a token either at an input buffer or at an output request flag of the successor switch.

The method used is essentially the same as before. The equations are as follows.

\[ p_{ij}^k = \text{probability \{ a packet exists either at an input buffer or has set an output request flag at stage } j \text{ in } \tau_k \} \]

\[ Z = \sum_{m=8-14,18-24} \frac{p^k_{m,j}}{2} + \sum_{m=4-7,15-17} \frac{1}{2} p^k_{m,j} \]
Figure 4.3.12 The distinguishable states of a (2 x 2) crossbar type basic switch for \( t_{\text{select}} = t_{\text{pass}} = t_{\text{delay}}/2 \).
\[ p_{oj}^k = \text{probability \{ a \ packet \ exists \ at \ an \ output \ link \ of \ a \ switch \ in \ stage \ j \ at \ time \ } t_k \} \]

\[ \approx \sum_{m=3,7,13,14,17,23,24} \frac{k}{p_{mj}} \sum_{m=2,5,6,10,11,12,16,20,21,22} \frac{1}{2} \frac{k}{p_{mj}}. \]

\[ p_{pass}^k = \text{probability \{ a \ packet \ exists \ either \ at \ and \ input \ buffer \ or \ has \ set \ an \ output \ request \ flag \ at \ stage \ j \ in } t_k \text{ and is passed \}} \]

\[ \approx p_{9,j}^k + \sum_{m=4,6,8,11,12,18,21} \frac{1}{2} \frac{k}{p_{mj}}. \]

\( p_j^k \) and \( p_j^k \) are defined as for case 1 and are computed using eqs. 4.3.21 and 22. State transitions are computed in two steps. In step 1 the transitions due to packet arrivals at stage 0 and packet removals at stage \( n-1 \) are carried out. In step 2 the state transitions due to other packet movement are computed. State transition tables and transition equations are shown in fig. 4.3.13. The throughput is estimated as,

\[ \text{NTP} \approx 2.0 \times p_{n-1}^k \text{ and } \text{tp} = \frac{2^n \cdot \text{NTP}}{t_{\text{delay}}}. \]

The TT is estimated as follows. Let
\( p_{18,0}^k = p_{18,0} + p_{4,0}^k \)
\( p_{19,0}^k = p_{19,0} + p_{1,0}^k + p_{15,0}^k \)
\( p_{20,0}^k = p_{20,0} + p_{5,0}^k \)
\( p_{21,0}^k = p_{21,0} + p_{6,0}^k \)
\( p_{22,0}^k = p_{22,0} + p_{2,0}^k + p_{16,0}^k \)
\( p_{23,0}^k = p_{23,0} + p_{7,0} \)
\( p_{24,0}^k = p_{24,0} + p_{17,0} + p_{3,0}^k \)
\( p_{m,0}^k = 0 \), for \( m = 1 - 7, 15 - 17 \).

(a) Step 1 at stage \( S_0 \):
\( p_{m,j}^k = p_{m,j}^k \) for \( 1 \leq m \leq 24, 0 \leq j \leq n-1 \)

(b) Step 1 at stage \( S_j \), for \( 0 < j < n-1 \):
\( p_{1,n-1}^k = p_{1,n-1}^k + p_{2,n-1}^k + p_{3,n-1}^k \)
\( p_{4,n-1}^k = p_{4,n-1}^k + p_{5,n-1}^k + p_{6,n-1}^k + p_{7,n-1}^k \)
\( p_{8,n-1}^k = p_{8,n-1}^k + p_{10,n-1}^k + p_{12,n-1}^k + p_{13,n-1}^k \)
\( p_{9,n-1}^k = p_{9,n-1}^k + p_{11,n-1}^k + p_{14,n-1}^k \)
\( p_{15,n-1}^k = p_{15,n-1}^k + p_{16,n-1}^k + p_{17,n-1}^k \)
\( p_{18,n-1}^k = p_{18,n-1}^k + p_{20,n-1}^k + p_{21,n-1}^k + p_{23,n-1}^k \)
\( p_{19,n-1}^k = p_{19,n-1}^k + p_{22,n-1}^k + p_{24,n-1}^k \)
\( p_{m,n-1}^k = 0 \), for \( m = 2 - 3, 5 - 7, 10 - 14, 16 - 17, 20 - 24 \).

(c) Step 1 at stage \( S_{n-1} \).

Figure 6.3.13 State transition tables for \( t_{pass} = t_{select} = 0.5 \).
Transition matrix \( T_j \)

\[
\begin{array}{cccccccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\
\hline
1 & q_j^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & q_j^2 \tilde{p}_j & q_j^2 \tilde{q}_j & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
3 & q_j^2 \tilde{p}_j & 2q_j^2 \tilde{p}_j \tilde{q}_j & q_j^2 \tilde{q}_j & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
4 & 0 & q_j & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
5 & 0 & 0 & 0 & q_j \tilde{p}_j & q_j \tilde{q}_j & 0 & 0 & 0 & 0 & 0 & 0 \\
6 & 0 & q_j \tilde{p}_j & q_j \tilde{q}_j & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
7 & 0 & 0 & 0 & q_j \tilde{p}_j & q_j \tilde{q}_j & q_j \tilde{p}_j \tilde{q}_j & q_j \tilde{q}_j & 0 & 0 & 0 & 0 \\
8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
9 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
11 & 0 & 0 & 0 & 0 & 0 & \tilde{p}_j & \tilde{q}_j & 0 & 0 & 0 & 0 \\
12 & 0 & 0 & 0 & 0 & \tilde{p}_j & 0 & \tilde{q}_j & 0 & 0 & 0 & 0 \\
13 & 0 & 0 & 0 & 0 & 0 & 0 & \tilde{p}_j^2 \tilde{q}_j & 0 & 0 & 0 & 0 \\
14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \tilde{p}_j^2 \tilde{q}_j & 0 & 0 & 0 \\
15 & 0 & 0 & 0 & q_j & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
16 & 0 & 0 & 0 & q_j \tilde{p}_j & 1 & q_j \tilde{p}_j & q_j \tilde{p}_j & q_j \tilde{p}_j & 0 & 0 & 0 \\
17 & 0 & 0 & 0 & q_j \tilde{p}_j & q_j \tilde{p}_j & q_j \tilde{p}_j & q_j \tilde{p}_j & q_j \tilde{p}_j & 0 & 0 & 0 \\
18 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\
19 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 \\
20 & 0 & 0 & 0 & q_j & 0 & 0 & 0 & 0 & \frac{1}{2} \tilde{p}_j & \frac{1}{2} \tilde{q}_j & \frac{1}{2} \tilde{p}_j & \frac{1}{2} \tilde{q}_j \\
21 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} \tilde{p}_j & \frac{1}{2} \tilde{p}_j & \tilde{q}_j & 0 & 0 & 0 \\
22 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} \tilde{p}_j & \frac{1}{2} \tilde{p}_j & \frac{1}{2} \tilde{q}_j & \frac{1}{2} \tilde{q}_j & \frac{1}{2} \tilde{p}_j & \frac{1}{2} \tilde{q}_j \\
23 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} \tilde{p}_j & \frac{1}{2} \tilde{p}_j & \frac{1}{2} \tilde{q}_j & \frac{1}{2} \tilde{q}_j & \frac{1}{2} \tilde{p}_j & \frac{1}{2} \tilde{q}_j \\
24 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} \tilde{p}_j & \frac{1}{2} \tilde{p}_j & \frac{1}{2} \tilde{q}_j & \frac{1}{2} \tilde{q}_j & \frac{1}{2} \tilde{p}_j & \frac{1}{2} \tilde{q}_j \\
\end{array}
\]

\( p_{i,j}^{k+1} = \sum_{m=1}^{26} p_m^{k} r_j^{m+1} \), for \( 1 \leq i \leq 24, 0 \leq j \leq n-1 \), \( k = 0, 1, \ldots \).

**Figure 4.3.13(d)** State transition tables, for step 2, at stage \( \tilde{s}_j \), with \( t_{\text{select}} = t_{\text{pass}} = 0.5 \).
### Transition matrix $\tau_j$ (continued)

<table>
<thead>
<tr>
<th></th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
<th>23</th>
<th>24</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>$2\bar{\tau}_j$</td>
<td>0</td>
<td>0</td>
<td>$\bar{\tau}_j^2$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>$2\bar{\tau}_j\bar{\tau}_j$</td>
<td>$2\bar{\tau}_j\bar{\tau}_j$</td>
<td>0</td>
<td>0</td>
<td>$\bar{\tau}_j^2\bar{\tau}_j$</td>
<td>0</td>
<td>0</td>
<td>$\bar{\tau}_j^2\bar{\tau}_j$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>$2\bar{\tau}_j\bar{\tau}_j^2$</td>
<td>$4\bar{\tau}_j\bar{\tau}_j\bar{\tau}_j$</td>
<td>$2\bar{\tau}_j\bar{\tau}_j^2$</td>
<td>0</td>
<td>0</td>
<td>$\bar{\tau}_j^2\bar{\tau}_j^2$</td>
<td>0</td>
<td>$2\bar{\tau}_j\bar{\tau}_j^2\bar{\tau}_j$</td>
<td>0</td>
<td>$\bar{\tau}_j^2\bar{\tau}_j^2$</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\bar{\tau}_j$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\bar{\tau}_j\bar{\tau}_j$</td>
<td>0</td>
<td>0</td>
<td>$\bar{\tau}_j\bar{\tau}_j$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\bar{\tau}_j\bar{\tau}_j^2$</td>
<td>$\bar{\tau}_j\bar{\tau}_j^2\bar{\tau}_j$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\bar{\tau}_j\bar{\tau}_j^2\bar{\tau}_j$</td>
<td>0</td>
<td>0</td>
<td>$\bar{\tau}_j\bar{\tau}_j^2\bar{\tau}_j$</td>
<td>0</td>
<td>$\bar{\tau}_j\bar{\tau}_j^2\bar{\tau}_j$</td>
<td>0</td>
<td>$\bar{\tau}_j\bar{\tau}_j^2\bar{\tau}_j$</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>$\bar{\tau}_j^2$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>$\bar{\tau}_j^2$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\bar{\tau}_j$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\bar{\tau}_j\bar{\tau}_j$</td>
<td>0</td>
<td>$\frac{1}{2}\bar{\tau}_j\bar{\tau}_j$</td>
<td>$\frac{1}{2}\bar{\tau}_j\bar{\tau}_j$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\bar{\tau}_j\bar{\tau}_j^2$</td>
<td>0</td>
<td>$\bar{\tau}_j\bar{\tau}_j\bar{\tau}_j$</td>
<td>0</td>
<td>$\frac{1}{2}\bar{\tau}_j\bar{\tau}_j^2\bar{\tau}_j$</td>
<td>0</td>
<td>$\bar{\tau}_j\bar{\tau}_j^2\bar{\tau}_j$</td>
</tr>
<tr>
<td>18</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>21</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>22</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>23</td>
<td>$\frac{1}{2}\bar{\tau}_j^2$</td>
<td>$\frac{1}{2}\bar{\tau}_j^2$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>24</td>
<td>$\frac{1}{2}\bar{\tau}_j^2$</td>
<td>$\frac{1}{2}\bar{\tau}_j^2$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

*Figure 4.3.13(d) Continued.*
\( p_{w_j}^k \) = probability \{ a packet corresponding to a token at an output request buffer exists in a switch at stage \( j \) in \( \tau_k \) \}

\[
\approx \frac{1}{2} \sum_{m=8}^{14} p_{m,j}^k + \sum_{m=4}^{18,20,21,23} \frac{1}{2} p_{m,j}^k .
\]

Define

\[
\mu_k p_{w_j}^k = \frac{p_{\text{pass}}^k}{\mu_j} .
\]

Observe that packets are delayed for a constant time in the \texttt{t\_select} phase. Then, as in eq. 4.3.16,

\[
TT \approx \frac{t_{\text{delay}}}{2} \sum_{j=0}^{n-1} \left( 1 + \frac{1}{p_{w_j}^k} \right) , \quad \text{and}
\]

\[
\text{NTT} \approx \frac{1}{2n} \sum_{j=0}^{n-1} \left( 1 + \frac{1}{p_{w_j}^k} \right) .
\]

The resulting performance in terms of NTP and NTT is graphed in figs. 4.3.9 and 10.

This model with 24 states can be used to estimate performance for other rational values of \texttt{t\_pass} and \texttt{t\_select}.
**Infinite Buffers**

Now suppose that there are infinite buffers between the stages of the delta network. For this case, the probability of a packet arriving at one of the input links of a switch at \( t_k \), \( k = 0, 1, \ldots \), is independent of a packet arrival at the other input link of the same switch at the same time. Each switch not in stage \( S_0 \) can be modelled by a two dimensional random walk as shown in fig. 4.3.14. We have not been able to obtain closed form solutions for this two dimensional chain. However, for the case of maximum loading under consideration, the NTP and NIT can be obtained as follows. For \( t_{\text{select}} = 0 \), \( NTP = p_{n-1} = 0.75 \) and for \( t_{\text{pass}} = 0 \), \( NTP = p_{n-1} = 1.0 \). These values follow directly by contradiction. Assuming them to be false leads to infinite queues for which case they are also true. They can also be shown by the following reasoning. A study of the two dimensional chain reveals that the dependence between the two queues occurs because of the different state change probabilities when one of the queues is empty (i.e. at the boundaries of the two dimensional chain). For heavy loads, the probability of a queue being empty is very small and can be assumed to be zero. The queues are then independent and the chain reduces to two unidimensional Markov chains shown in fig. 4.3.15 which is the same as the chain in section 3.3. This chain was solved in section 3.3 and the results were,

\[
p_{n_0} = \frac{p_a - p_i}{p_a}, \quad TT = \frac{t_{\text{slot}} (1 - p_i)}{p_a - p_i} \quad \text{for} \quad p_i < p_a.
\]
(a) A switch with infinite buffers

(b) Markov model for switch in (a) with probability of arrival of a packet in t_pass of $p$.

**Figure 4.3.14** Modelling delta networks with infinite buffers.
Figure 4.3.15
Markov chain for an input buffer with Bernoulli inputs and assumed $p_a$.
Thus, \( \lim_{n \to 0} p_n = 0 \), which agrees with the hypothesis. As \( p_i \to p_a \),
\[ p_i \to p_a \]
TT \( \to \infty \), while NTP \( \to p_a \) (since a queue is never empty). For
t\_select = 0, NTP = \( p_a = 0.75 \) and for t\_pass = 0, NTP = \( p_a = 1.0 \), as was
cconcluded earlier. NTP for this case is also shown in fig. 4.3.9.

4.3.1.2. Simulation

In this section we present simulation results for \((2^N x 2^N)\) delta networks which substantiate the analysis of the preceding section and extend the results to analytically intractable cases.

We have developed a general event driven simulator for multistage interconnection networks in a packet communication environment. This simulator can handle several switch types, inter-stage interconnection patterns, loading conditions, and switch operation policies. We also have used several special simulators for specific interconnections for time and space efficiency.

The performance measures used are those of average throughput and turn-around-time and these measures normalized with respect to their maximum possible values. Regenerative simulation is not being used currently because the large number of distinguishable states makes a regeneration point a rare event that is hard to detect. The point estimates reported are obtained by gathering statistics after a specifiable settling time and averaging over a large number of packets (typically 5000 to 20,000) passing through the network.
There are a large number of topologically distinguishable \(2^n \times 2^n\) delta networks, for any constant \(n\). The approximate analysis of the preceding section is valid for all delta networks leading to the hypothesis that all delta networks of the same size have the same performance. Simulation experiments were carried out to test this hypothesis. Several topologically different \(2^n \times 2^n\) buffered delta networks, \(3 \leq n \leq 6\), were selected and simulated. It was found that, within simulation accuracy, the performance of a \(2^n \times 2^n\) delta network was indistinguishable from the performance of topologically different networks with the same value of \(n\). These experiments were limited to only a few selected values of the parameters listed below. Extensive simulation was carried out for omega networks [Lawrie75a] (excluding the permutation at the input of the network) and these results are reported in this section.

The following parameters were varied in simulation runs:

(i) the size of the network.

(ii) \(t_{\text{select}}\) and \(t_{\text{pass}}\) with \((t_{\text{select}} + t_{\text{pass}}) = 1\).

(iii) the number of buffers between stages.

The maximum throughput is that for an ideal network with no conflicts, and is estimated as \(\text{max}_t = (2^n/(t_{\text{select}} + t_{\text{pass}}))\). The minimum delay is \(\text{min}_d = n.(t_{\text{select}} + t_{\text{pass}})\). The throughput and turn around time are normalized with respect to these bounds, and denoted by NTP and NTT respectively.
The variation of NTP and NTT with the number of stages and with t_select and t_pass is shown in figs. 4.3.16-19. Also shown on figs. 4.3.16 and 17 are the first and second order estimates of the preceding section. The second order analytical estimates compare very well with simulation estimates. The difference between these estimates is less than 5% for the range simulated. The first order (binomial) approximation compares well with the other estimates for networks with a small number of stages the difference being less than 10% for networks with upto five stages. Further comparison and discussion appears in the next section.

The variation of the normalized thruput and turn around time with buffer size and with the number of stages as parameter is shown in figs. 4.3.20 and 21. The average thruput approaches the predicted asymptotes for large buffer sizes. However, the turn around time for large buffer lengths is very poor. This agrees with the result in the previous section that for infinite buffers between stages the TT is infinite at the maximum load.

4.3.1.3. Discussion and Comparison

In this section we discuss the results obtained in the preceding sections. We also compare the performance of buffered and unbuffered networks and motivate the study of model (ii), to be presented in the following section.
Figure 4.3.16  NTP versus the number of stages (n).
(Single buffer between stages.)
Figure 4.3.17  NTT versus the number of stages (n).
(Single buffer between stages.)
Figure 4.3.18  Normalized throughput versus t_select (= 1 - t_pass).
(Single buffer between stages.)
Figure 4.3.19  Normalized turn around time versus \( t_{\text{select}} = 1.0 - t_{\text{pass}} \).
(Single buffer between stages.)
Figure 4.3.20  Normalized throughput versus the number of buffers between stages. \((t_{\text{select}} + t_{\text{pass}} = 1.0)\)
Figure 4.3.21  Normalized turn around time versus the number of buffers between stages. \( (t_{\text{select}} + t_{\text{pass}} = 1.0) \)
We first compare the performance of buffered with unbuffered networks. The NTP for the various networks is shown in fig. 4.3.22. (We use the second order analysis in this figure.) For all the networks, except delta networks with infinite buffers, NTP decreases monotonically with increase in the number of stages (n). For large crossbar switches NTP converges to a constant. For buffered delta networks NTP decreases very slowly as the size of the networks become large. The rate of fall of NTP is much faster for unbuffered delta networks than for the buffered case. However, for the range of practical values of n, NTP of the unbuffered networks is comparable to that of the buffered networks.

The important point to note is that the pipelining effect in buffered delta networks produces a substantial improvement in throughput. This can be seen as follows. The maximum (conflict free) throughput can be expressed as \((2^n / t_{slot})\). Let \(t_{delay}\) be the minimum delay that a packet encounters at a stage of the networks. Then, \(t_{slot} > \min_d = n \times t_{delay}\) for the unbuffered delta network composed of fixed size basic switches. For the buffered delta network and the full crossbar switch, \(t_{slot} = t_{delay}\). Thus, there is a factor of n difference in \(t_{slot}\) and hence in \(\text{MAXTP}\). This factor of n difference in \(t_{slot}\) is used in computing the speedup due to buffering and results are graphed in fig. 4.3.23. This figure indicates that for most values of buffer length and \(t_{select}\), the speedup is more than a factor of n. For \(t_{select}\) of zero, and a single buffer between stages, the speedup is less than a factor of n. Poor performance for this case is the motivation for considering model (ii). For this value of \(t_{select}\), the
Figure 4.3.22  NTP versus the number of stages for various networks.
Figure 4.3.23  Speedup due to buffering.
speedup can be improved by increasing the number of buffers between stages.

\[ \text{delay for an } (N \times N) \text{ crossbar switch is } \Omega(\log N) \text{ (i.e. it is asymptotically lower bounded by this function)} \ [\text{Kuck78a; Horowitz78a}] \]
while for a single \((2 \times 2)\) switch it is a constant. Thus, the buffered delta network has a better throughput than the simple crossbar switch for large networks. However, the throughput of the crossbar can be improved by internal buffering within the switch. The main asset of delta networks is that for an \((N \times N)\) network, the number of gates required is \(O(N \log N)\) while \((N \times N)\) crossbar switches require \(O(N^2)\) gates.

We next compare TT and NTT for buffered and unbuffered networks. The minimum delay that a packet encounters in both the buffered and unbuffered networks is \(O(\log N)\) for an \((N \times N)\) network. However, for unbuffered delta networks with fixed size basic switches, NTT (estimated by \(T_a\)) increases almost linearly with increases in the number of stages as shown in fig. 3.3.1. For buffered delta networks with a single buffer between stages, NTT reaches a maximum and then decreases as shown in fig. 4.3.10. This is because the latter stages of the buffered delta network are never heavily loaded and do not cause much packet delay. Comparison of the figures shows that the actual values of NTT are less for buffered delta networks than in unbuffered networks for any \(n\). Thus buffering also produces an improvement in TT.

We next consider the performance obtained with variation in \(t_{\text{select}}\) and \(t_{\text{pass}}\) as shown in figs. 4.3.16-20. The best performance
(maximum TP and minimum TT) is obtained for \( t_{\text{select}} = 1 \) and \( t_{\text{pass}} = 0 \).
The TP decreases monotonically with decrease in \( t_{\text{select}} \) for constant \( t_{\text{select}} + t_{\text{pass}} \). For large networks, a second order interpolation between the analytically estimated values produces a good estimate for other values of this parameter. For large networks, the NTT varies almost linearly between the extreme choices of \( t_{\text{select}} \) and \( t_{\text{pass}} \).
This characteristic (fig. 4.3.19) for small networks is interesting. There are local minima in the NTT at \( t_{\text{select}} = 0, 0.5, \) and 1. This is because there is better pipelining for these values. For larger networks this effect gets swamped by delays due to conflicts in the network.

The routing algorithm used in these networks is very simple. Hence time \( t_{\text{select}} \) in an implementation will be rather small. The ratio of \( t_{\text{select}} \) to \( t_{\text{pass}} \) depends upon the size of the packets and on the fraction of the packet that is transferred in parallel. For small packets passed along links in parallel we can expect \( t_{\text{select}} \) and \( t_{\text{pass}} \) to be comparable. For larger packets and/or serial transfer along links, \( t_{\text{select}} \) will be much less than \( t_{\text{pass}} \). Thus in most implementations we can anticipate operation in the lower performance areas of the characteristics of figs. 4.3.18,19.

Now consider the performance with variation in the buffer size. The rate of increase in throughput with the number of buffers is largest for a change from one to two buffers and this rate of increase falls rapidly as the buffer size increases. Further, fig. 4.3.21 shows that
the NTT increases almost linearly with increase in the buffer size. Thus, it seems that, for most applications, the number of buffers between stages should be limited to one or two.

Observe that for \( t_{\text{select}} = 0 \) there is a significant increase in the throughput for an increase in the buffer size from one to two. In fact, for \( t_{\text{select}} = 0 \) and for large networks the throughput almost doubles for an increase in the buffer size from one to two. For \( t_{\text{select}} = 1 \), the increase in throughput in increasing the buffer length to two from one, is not as significant as for the preceding case. Further, for both cases, TT almost doubles with this change in buffer length.

The increase in TP in the change from one to two buffers for \( t_{\text{select}} = 0 \) can be attributed to better pipelining in the network. Recall that for the present model, a packet is not advanced in a \( t_{\text{pass}} \) interval until a buffer becomes available at the successor buffer. Suppose that the successor buffer contains a packet that is advanced in a \( t_{\text{slot}} \) interval, the model with a single buffer does not allow the buffer to be filled in the same \( t_{\text{slot}} \) interval. With two buffers this becomes possible and this is what we have referred to as the "pipelining effect". The concomitant increase in the TT is because a packet is often second in a first in first out queue at a switch input and thus waits longer at a switch.

We observed earlier that in practice we can expect \( t_{\text{select}} \) to be less than \( t_{\text{pass}} \). For large packet sizes and/or serial transfer of packets between switches, \( t_{\text{select}} \ll t_{\text{pass}} \). For this case we have noted
that TP almost doubles in increasing the buffer size from one to two, but that TT doubles as well. This is the motivation for model (ii) that we consider next. We want to obtain better pipelining for t_select \ll t_pass but without paying in terms of an increase in the TT.

4.3.2. Model (ii)

We now consider some modifications to model (i) that result in better performance for t_select \ll t_pass. We noted in the previous section that increasing the buffer length from one to two in model (i) leads to better pipelining and better throughput. However, the addition of a second buffer approximately doubles TT at maximum load. The reason is that at maximum load, there are often two packets in a buffer and this leads to longer waits. The modification we now consider is to add on another buffer to allow for better pipelining. This buffer is called the "hold" buffer. When the hold buffer is full, the buffer from which the packet arrived is not released. As we will see, this leads to operation similar to the case of t_pass = 0 which was the upper end of performance of model (i).

A timed Petri net model for the operation of a (2 x 2) switch in this model is shown in fig. 4.3.24. Comparison with fig. 4.3.1 reveals that the difference in model (ii) is the addition of the hold buffers. Whenever the hold buffer is empty and a packet arrives at an output request flag or if one is waiting at an output request flag, a transfer to the hold buffer is initiated. The hold buffer is cleared and the
Figure 4.3.24  Timed Petri net model for the operation of a (2 x 2) basic switch in model (II) (shown in the initial state for a single buffer between stages).
successor buffer is filled as soon as the successor buffer becomes empty. When the packet is in the hold buffer, the input buffer from which the packet was passed is not freed. This buffer is freed as soon as the packet is advanced from the hold buffer into the successor buffer.

The idea behind this mode of operation is to have the switch operate in an identical manner for both the extreme cases of \( t_{\text{select}} = 0 \) and of \( t_{\text{pass}} = 0 \). In model (i) we noted that for large networks, the ratio of TP and TT at the extreme values of these parameters differed by a factor of two. This difference is eliminated in model (ii). For the case of \( t_{\text{select}} = 1.0 \) and \( t_{\text{pass}} = 0.0 \), the operation is identical to that of case 2 of the previous section, and the performance characteristics are also identical. For the other extreme case of \( t_{\text{select}} = 0 \) and \( t_{\text{pass}} = 1 \), the operation at a switch is again the same as the previous case except at the last stage of the network. For \( t_{\text{pass}} = 0 \), two packets can be put out at the same output in the same time slot because of the assumption that packets at the output are removed instantaneously. This lead to the elimination of the last stage of the network from the analysis in the previous section, case 2. For \( t_{\text{pass}} = 1 \), at most one packet can be put out at an output link in the same \( t_{\text{slot}} \) interval. The result is that TP for an \( n \) stage delta network with \( t_{\text{pass}} = 1 \) is the same as TP for an \( (n+1) \) stage delta network with \( t_{\text{pass}} = 0 \). We observed in the last section that for large networks, TP decreases very slowly for increase in \( n \). Thus, for large networks TP is almost the same at these extremes. TT can also be
obtained by similar reasoning.

The characteristics for this model are shown in figs. 4.3.25-29. The second order estimate is seen to be very good in the range for which simulations were carried out. The shape of the NTP and NTT versus n characteristics is similar to that for model (i) and the comments of section 4.3.1.3 apply to model (ii) as well. The figures show that there is little difference in TP and TT for large networks of the same size with different values of t_select and t_pass for constant t_select + t_pass. Comparing NTT for t_select = 0 for model (ii) (fig. 4.3.26) with that for model (i) (fig. 4.3.27) shows about a two-fold improvement. This factor of improvement is also observed for the same size buffer lengths (figs. 4.3.21 and 29).

In summary, model (ii) produces a considerable improvement in the TT for small values of t_select. For t_select = 0, the TP for model (i) with two buffers is almost identical to the TP for model (ii) with one hold buffer and one additional buffer. However, there is greater than a two-fold improvement in the TT in using model (ii) for this case.

We now present some simulation results, for the network performance of model (ii), at lower than the maximum load. Specifically, packets are assumed to arrive with an exponential interarrival time distribution, with mean of $\beta$ time units, after a buffer at a network input link becomes available. Thus, the maximum load corresponds to a $\beta$ of zero. As before, let MAXTP denote the ideal (conflict free) throughput at maximum load. Denote by MAXTP2, the ideal (conflict free) throughput at any value
Figure 4.3.25  Normalized throughput (NTP) versus the number of stages (n) for model (II).
(Single buffer between stages.)
Figure 4.3.26 Normalized turn around time (NTT) versus the number of stages (n), for model (ii). (Single buffer between stages.)
Figure 4.3.27  Normalized throughput versus $t_{select} = 1 - t_{pass}$, for model (11).

(Single buffer between stages.)
Figure 4.3.28  Normalized turn around time (NTT) versus $t_{select} = 1 - t_{pass}$, for model (ii).
(Single buffer between stages.)
Figure 4.3.29  NTP and NTT versus the number of buffers between stages for model (ii).  (t_select = 0;  t_pass = 1.)
of beta. Thus, MAXTP2 equals maxtp for β of zero. MAXTP2 is given by
\[ \frac{2^n}{t_{\text{select}} + t_{\text{pass}} + \beta} \]. This is the average input rate if no conflicts occur in the network. The throughput normalized with respect to MAXTP and MAXTP2 is denoted by NTP and NTP2 respectively.

Fig. 4.3.30 shows the variation of NTP, NTP2, and NTT with β. First, observe that NTP decreases monotonically with increase in β. This means that the networks exhibit throughput stability. Second, note that NTP2 increases rapidly with increase in β and quickly approaches unity. Third, NTT first decreases fairly fast with increase in β and then decreases relatively slowly for larger β. Similar characteristics are also obtained for model (i).

4.5. (b^n x b^n) Delta networks.

In the previous section we considered the performance of (2^n x 2^n) delta networks. We now briefly consider the performance of (b^n x b^n) delta networks with b greater than two. These networks are constructed from (b x b) basic switches. We make the assumptions stated in section 4.2.

First, note that packets are routed through a (b^n x b^n) delta network using base b digit controlled routing (as described in chapter 2). In such a network, network output links are labelled by n-digit base b numbers and switch output links have a base b digit label. Considering the binary internal representation in current implementations, b will
Figure 4.3.30  NTP and NTT versus av. interarrival time ($\beta$).
($t_{\text{select}} = 0; t_{\text{pass}} = 1; \text{Model (ii); single buffer.}$)
normally be restricted to be a power of two. Thus, we assume in the following that \( b = 2^k \) for some positive integer \( k \).

We now make two important observations. First, recall that in the preceding section we observed that buffering produced a greater than \( n \)-fold improvement in the throughput as compared to that of the corresponding unbuffered delta network. This was because of two effects. Buffering produces a pipelining effect so that there is stage wise parallelism in packet transfer in addition to parallel transfer within a stage. Secondly, there is an increase in NTP because of fewer conflicts in the network. This suggests that we should have as many stages as possible with buffers between stages. This leads to the conclusion that we should construct the network from \((2 \times 2)\) switches (which, of course, is the smallest size switch leading to the maximum number of stages in the network).

Recall from chapter 2 that the basic switches of a delta network can be replaced by sub-delta networks to obtain a delta network with smaller basic switches. Algorithm 2.2.1 specifies how this can be done. Thus the \((b \times b)\) basic switches in a \((b^n \times b^n)\) delta network with \( b = 2^k \) can be replaced by \((2^k \times 2^k)\) delta networks to obtain a \((2^{n \cdot k} \times 2^{n \cdot k})\) delta network composed of \((2 \times 2)\) basic switches. These observations can be summarized as follows.
(i) For \((b^n \times b^n)\) delta networks the case of practical interest has \(b = 2^k\) for \(k = 1, 2, \ldots\).

(ii) For this case, the \((b \times b)\) basic switches can be replaced by \((2^k \times 2^k)\) delta networks to yield a \((2^{n \cdot k} \times 2^{n \cdot k})\) delta network composed of \((2 \times 2)\) switches.

(iii) Buffering between each of the stages of the resulting \((2^{n \cdot k} \times 2^{n \cdot k})\) network produces better than a \((n \cdot k)\) fold improvement in the throughput.

This says that, if the basic switches are constructed as smaller delta networks, then it would be advantageous to have internal buffering between the stages of these smaller delta networks. The results of the preceding section can then be used to estimate the network performance.

The \((b \times b)\) basic switches may be available as VLSI chips. Pin limitations will then restrict the size of the basic switches. Dennis [Dennis80a] and his co-workers at MIT have proposed building a \((4 \times 4)\) basic switch as a 100 pin LSI chip. This seems to be the maximum feasible size basic switch that can be constructed as a single LSI/VLSI chip with current technology because of the above-mentioned pin limitations.

We now briefly look at the case when the basic switches are constructed as crossbar switches. The binomial approximation of the previous section can be used for this case. In fact all equations in that section were developed with parameter \(b\) and are valid for this case. The performance characteristics using model (i) are shown in
In the previous section we ascertained by comparison with the second order model and with simulation results that the binomial approximation produced good estimates of NTP and NTT (within 5%) for \((2^n \times 2^n)\) delta networks with less than 6 stages. For \(b\) greater than two the second order approximation has too many states to be analytically tractable. Neither do we have simulation results. The performance characteristics reported here should hence be used with caution. They can be expected to be good for a small number of stages but should be used to look for trends in performance rather than exact performance values.

The trends indicated by these characteristics are as follows. For large basic switch sizes NTP and NTT seem to depend only on the number of stages in the network and are relatively insensitive to increases in the basic switch size. The shape of NTP and NTT characteristics is similar for different values of \(b\). The rate of decrease in NTP and the rate of increase in NTT fall off rapidly with increase in the number of stages. Of principal interest are the cases of basic switch sizes of two and four. It is observed that as the network size increases, the difference in NTP and NTT for these cases is minimal.

4.5. Pruned and Augmented Delta Networks

In this section we extend our study to buffered pruned and augmented delta networks. We first present analysis techniques and results for the performance of arbitration networks. As discussed in chapter 2,
Figure 4.4.2
NTP versus log₂ N
(Binomial approximation)

Figure 4.4.1
NTP versus n
(Binomial approximation)
Figure 4.4.3  NTT versus n (binomial approximation).

Figure 4.4.4  NTT versus $\log_2 N$ (binomial approximation).
these are \( n \)-stage, \((b^n \times b^m)\) networks where \( n > m \), \( m \) stages are composed entirely of square switches, and \( n-m \) stages contain only \((b \times 1)\) arbitration switches. Similar analysis techniques and results can also be obtained for distribution networks \((b^n \times b^m)\) networks with \( m > n \). In particular we will consider the case of \( b = 2 \). In the previous section we have discussed why this case is the most interesting one. Finally, we briefly discuss the performance of buffered augmented delta networks.

There are several ways to construct a \((b^n \times b^m)\) regular arbitration network. This was discussed in detail in chapter 2. To recapitulate, the \( m \) square stages and the \( n-m \) arbitration stages can be arranged in any order. However, the arrangement chosen will affect both the size of the network (i.e., the number of switches it contains) and its performance. For any arbitration network, there are two extreme choices for the arrangement of the stages. In one, all the arbitration stages are grouped together at the front of the network. In the other, the arbitration stages are grouped together and located after all the square stages. These two possibilities are shown in fig. 4.5.1 for \((2^3 \times 2^1)\) arbitration networks. Note that the first choice, with the arbitration stages at the front of the network, will have fewer switches than the other choice. This is because the total number of switches in arbitration stages will not change with the location of the stage, but the number of switches in a square stage is less if it follows arbitration stages. In a \((2^n \times 2^m)\) arbitration networks there are \( \binom{n}{m} - 2 \) other choices of arbitration networks with intermediate numbers of switches.
Figure 4.5.1  Extreme arbitration networks.
between these extremes.

We first consider the case in which the switches in all of the stages have equal delay. The analysis technique of the preceding sections can be extended to arbitration networks. We will present results of a second order analysis analogous to that in section 4.3. To this end we identify the distinguishable states of a (2 x 1) switch and these are shown in fig. 4.5.2. The technique and the notation used is the same as in section 4.3. The equations and transition tables derived in that section are valid at square stages. At arbitration stages the following equations and transition tables hold.

\[ p_{ij}^k = \sum_{m=3}^{4} \frac{1}{2p_{m,j}} + \sum_{m=5}^{6} \frac{k}{p_{m,j}} \]  \hspace{1cm} (4.5.1)

\[ p_{0j}^k = \sum_{m=2,4,6} \frac{k}{p_{m,j}} \]  \hspace{1cm} (4.5.2)

\[ p_{\text{pass}j}^k = \sum_{m=3,5} \frac{1}{2p_{m,j}} \]  \hspace{1cm} (4.5.3)

\[ p_{\text{fill}j}^k = \sum_{m=3,5} \frac{k}{p_{m,j}} \]  \hspace{1cm} (4.5.4)

The equations for \( p_{ij}^k \) and \( p_{0j}^k \) are the same as in section 4.3 for the corresponding cases. The difference is that the type of succeeding and previous stages (i.e., whether they are arbitration or square stages)
Figure 4.5.2 States of a (2 x 1) arbitration switch.
now determines the values of these parameters. The transition tables and equations for the cases of \( t_{\text{select}} = 0 \) and 1 (with \( t_{\text{pass}} = 1 - t_{\text{select}} \)) are shown in figs. 4.5.3 and 4.

The results obtained from this analysis are as follows. For the case in which the switches in all of the stages have the same delay, the arrangement of stages in buffered arbitration networks affects the performance in the same way as for unbuffered networks as discussed in section 3.4. That is, the best throughput is obtained with networks that place the arbitration stages after the square stages and this corresponds to the extreme network with the most switches, while the smallest networks and the worst performance result from placing all of the arbitration stages at the beginning of the network.

One way to balance the load in an arbitration network is to match the throughput of the individual stages by decreasing the delay of the arbitration stages to compensate for the reduction in paths. Thus, a stage of \((b \times 1)\) arbitration switches should have a delay \(\frac{1}{b}\) times the delay of the preceding stages and the delay of a stage of square switches should equal the delay of the preceding stage. This could be achieved by implementing a serial to parallel transformation on the information in the packets as they pass through an arbitration stage. Thus, a packet that enters an arbitration switch organized as \(k\) serially transmitted subpackets would leave the stage as \(\frac{k}{b}\) serially transmitted subpackets that are \(b\) times as large as the entering subpackets.
Transition matrix $T_j^1$

$$
\begin{array}{cccccc}
1 & 2 & 3 & 4 & 5 & 6 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 \\
2 & 0 & \bar{p}_j & \bar{q}_j & 0 & 0 & 0 \\
3 & 0 & 0 & 1 & 0 & 0 & 0 \\
4 & 0 & 0 & \bar{p}_j & \bar{q}_j & 0 & 0 \\
5 & 0 & 0 & 0 & 0 & 1 & 0 \\
6 & 0 & 0 & 0 & 0 & \bar{p}_j & \bar{q}_j \\
\end{array}
$$

$p_{1,j}^k = \sum_{m=1}^{6} p_{m,j}^k T_j^1(m,i)$, for $1 \leq i \leq 6$, $k = 0, 1 \ldots$

(a) State transition matrix for step 1.

Transition matrix $T_j^2$

$$
\begin{array}{cccccc}
1 & 2 & 3 & 4 & 5 & 6 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 \\
2 & 0 & 1 & 0 & 0 & 0 & 0 \\
3 & 0 & 1 & 0 & 0 & 0 & 0 \\
4 & 0 & 0 & 1 & 0 & 0 & 0 \\
5 & 0 & 0 & 0 & 1 & 0 & 0 \\
6 & 0 & 0 & 0 & 0 & 1 & 0 \\
\end{array}
$$

$p_{2,j}^k = \sum_{m=1}^{6} p_{m,j}^k T_j^2(m,i)$, for $1 \leq i \leq 6$, $k = 0, 1 \ldots$

(b) State transition matrix for step 2.

Transition matrix $T_j^3$

$$
\begin{array}{cccccc}
1 & 2 & 3 & 4 & 5 & 6 \\
1 & q_j^2 & 0 & 2p_j q_j & 0 & p_j^2 & 0 \\
2 & 0 & \bar{q}_j^2 & 0 & 2p_j \bar{q}_j & 0 & \bar{p}_j^2 \\
3 & 0 & 0 & \bar{q}_j & 0 & \bar{p}_j & 0 \\
4 & 0 & 0 & 0 & \bar{q}_j & 0 & \bar{p}_j \\
5 & 0 & 0 & 0 & 0 & 1 & 0 \\
6 & 0 & 0 & 0 & 0 & 0 & 1 \\
\end{array}
$$

$p_{3,j}^k = \sum_{m=1}^{6} p_{m,j}^k T_j^3(m,i)$, for $1 \leq i \leq 6$, $k = 0, 1 \ldots$

(c) State transition matrix for step 3.

Figure 4.5.3 State transition matrices at an arbitration stage $S_j$

$0 \leq j \leq \omega - 1$, for $t_{select} = 1 - t_{pass} = 1$ (model (i), single buffer between stages).
\[ p_{5,j,0}^k = p_{1,j,0}^k + p_{3,j,0}^k + p_{5,j,0}^k \]
\[ p_{6,j,0}^k = p_{2,j,0}^k + p_{4,j,0}^k + p_{6,j,0}^k \]
\[ p_{m,j,0}^k = 0, \text{ for } m = 1 - 4 \]

(a) Step 1 at an arbitration stage \( S_0 \).

\[ p_{m,j}^k = p_{m,j}^k, \text{ for } 1 \leq m \leq 6, \ 0 < j < n-1 \]

(b) Step 1 at an arbitration stage \( S_j \), for \( 0 < j < n-1 \).

\[ p_{1,j,n-1}^k = p_{2,j,n-1}^k + p_{3,j,n-1}^k \]
\[ p_{3,j,n-1}^k = p_{4,j,n-1}^k + p_{5,j,n-1}^k \]
\[ p_{5,j,n-1}^k = p_{6,j,n-1}^k + p_{5,j,n-1}^k \]
\[ p_{m,j,n-1}^k = 0, \text{ for } m = 2, 4, 6 \]

(c) Step 1 at an arbitration stage \( S_{n-1} \).

Transition matrix \( T_j^\beta \)

\[
\begin{array}{cccccc}
1 & 2 & 3 & 4 & 5 & 6 \\
\hline
1 & q_j^2 & 0 & 2p_j & 0 & q_j^2 & 0 \\
2 & q_j^2 p_j & q_j^2 p_j & 2p_j & 0 & q_j^2 p_j & q_j^2 p_j \\
3 & 0 & q_j & 0 & p_j & 0 & 0 \\
4 & 0 & 0 & q_j & q_j^2 & 0 & p_j & q_j^2 \\
5 & 0 & 0 & 0 & 1 & 0 & 0 \\
6 & 0 & 0 & 0 & 0 & q_j & q_j^2 \\
\end{array}
\]

\[ p_{i,j}^{k+1} = \sum_{m=1}^{6} p_{m,j}^k T_j^\beta(m,i), \text{ for } 1 \leq i \leq 6, \ k = 0, 1 \ldots \]

(d) Step 2 at an arbitration stage \( S_j \), for \( 0 \leq j \leq n-1 \).

Figure 4.5.4 State transition tables at an arbitration stage, for \( t_{\text{select}} = 1 - t_{\text{pass}} = 0 \) (model (1), single buffer between stages).
The analysis technique can be modified to handle this mode of operation of the networks. The cases analysed are again those with \( t_{\text{select}} = 0 \) and \( t_{\text{pass}} = 0 \) with \( (t_{\text{select}} + t_{\text{pass}} = t_{\text{delay}}) \). Here, the delay is not the same at different stages of the network and the delay at stage \( j \) is denoted by \( t_{\text{delay}}_j \). The time slot in the analysis is taken as the smallest delay at any stage (i.e., the smallest \( t_{\text{delay}}_j \) for \( 0 \leq j < n \)). For an arbitration network this is the delay at stage \( S_{n-1} \). Packet movement at stage \( j \) occurs in time intervals of \( t_{\text{delay}}_j \) and the switches are said to fire. For arbitration (and distribution) networks constructed as described above, \( t_{\text{delay}} \) is a multiple of \( t_{\text{slot}} \). Thus packet movement at all stages does not occur in every time slot but does occur at time instants that are multiples of the time slot. The equations and technique are the same as before except that \( p_{\text{fill}} \) and \( p_{\text{pass}} \) at stages that do not fire in a particular time slot are taken to be zero.

This analysis yields the following results. If the matching of stages described above is implemented, then the arrangement of stages that gives the best throughput is not necessarily one of the two extreme configurations shown in fig. 4.5.1. Some examples of optimal stage arrangements with the throughput as the criterion of optimality are given in figs. 4.5.5 and 6. Here, the arrangement of stages that give the highest normalized throughput is given as a string of A's and S's where A denotes an arbitration stage and S a square stage. These were obtained by using the above analysis technique to compute the normalized throughput.
<table>
<thead>
<tr>
<th>Network Shape</th>
<th>( t_{\text{select}=0} )</th>
<th>( t_{\text{select}=1} )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Arrangement</td>
<td>NTP</td>
</tr>
<tr>
<td>( 2^1 \times 2^0 )</td>
<td>A</td>
<td>1.00</td>
</tr>
<tr>
<td>( 2^1 \times 2^1 )</td>
<td>S</td>
<td>0.75</td>
</tr>
<tr>
<td>( 2^2 \times 2^0 )</td>
<td>AA</td>
<td>0.75</td>
</tr>
<tr>
<td>( 2^2 \times 2^1 )</td>
<td>SA</td>
<td>0.56</td>
</tr>
<tr>
<td>( 2^2 \times 2^2 )</td>
<td>SS</td>
<td>0.38</td>
</tr>
<tr>
<td>( 2^3 \times 2^0 )</td>
<td>AAA</td>
<td>0.75</td>
</tr>
<tr>
<td>( 2^3 \times 2^1 )</td>
<td>AAS</td>
<td>0.50</td>
</tr>
<tr>
<td>( 2^3 \times 2^2 )</td>
<td>SSA</td>
<td>0.42</td>
</tr>
<tr>
<td>( 2^3 \times 2^3 )</td>
<td>SSS</td>
<td>0.31</td>
</tr>
<tr>
<td>( 2^4 \times 2^0 )</td>
<td>AAAA</td>
<td>0.75</td>
</tr>
<tr>
<td>( 2^4 \times 2^1 )</td>
<td>AAAS</td>
<td>0.50</td>
</tr>
<tr>
<td>( 2^4 \times 2^2 )</td>
<td>SAAS</td>
<td>0.45</td>
</tr>
<tr>
<td>( 2^4 \times 2^3 )</td>
<td>SSAAS</td>
<td>0.36</td>
</tr>
<tr>
<td>( 2^4 \times 2^4 )</td>
<td>SSSS</td>
<td>0.27</td>
</tr>
<tr>
<td>( 2^5 \times 2^0 )</td>
<td>AAAA</td>
<td>0.75</td>
</tr>
<tr>
<td>( 2^5 \times 2^1 )</td>
<td>AAAS</td>
<td>0.50</td>
</tr>
<tr>
<td>( 2^5 \times 2^2 )</td>
<td>SAAS</td>
<td>0.48</td>
</tr>
<tr>
<td>( 2^5 \times 2^3 )</td>
<td>SSAAS</td>
<td>0.42</td>
</tr>
<tr>
<td>( 2^5 \times 2^4 )</td>
<td>SSSSA</td>
<td>0.32</td>
</tr>
<tr>
<td>( 2^5 \times 2^5 )</td>
<td>SSSS</td>
<td>0.25</td>
</tr>
<tr>
<td>( 2^6 \times 2^0 )</td>
<td>AAAAAA</td>
<td>0.75</td>
</tr>
<tr>
<td>( 2^6 \times 2^1 )</td>
<td>AAAAAS</td>
<td>0.50</td>
</tr>
<tr>
<td>( 2^6 \times 2^2 )</td>
<td>SAAAAS</td>
<td>0.50</td>
</tr>
<tr>
<td>( 2^6 \times 2^3 )</td>
<td>SSAAAAS</td>
<td>0.47</td>
</tr>
<tr>
<td>( 2^6 \times 2^4 )</td>
<td>SSAAAS</td>
<td>0.41</td>
</tr>
<tr>
<td>( 2^6 \times 2^5 )</td>
<td>SSASS</td>
<td>0.30</td>
</tr>
<tr>
<td>( 2^6 \times 2^6 )</td>
<td>SSSSSS</td>
<td>0.24</td>
</tr>
</tbody>
</table>

**Figure 4.5.5.** Optimal (buffered) arbitration networks (model (i), single buffer between stages).
<table>
<thead>
<tr>
<th>Network Shape</th>
<th>( t_{\text{select}}=0 )</th>
<th>( t_{\text{select}}=1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Arrangement</td>
<td>NTP</td>
</tr>
<tr>
<td>( 2^7 \times 2^0 )</td>
<td>AAAAAAA</td>
<td>0.75</td>
</tr>
<tr>
<td>( 2^7 \times 2^1 )</td>
<td>AAAAAAS</td>
<td>0.50</td>
</tr>
<tr>
<td>( 2^7 \times 2^2 )</td>
<td>SAAAAAS</td>
<td>0.50</td>
</tr>
<tr>
<td>( 2^7 \times 2^3 )</td>
<td>SSAAAAAS</td>
<td>0.49</td>
</tr>
<tr>
<td>( 2^7 \times 2^4 )</td>
<td>SSAAAAAS</td>
<td>0.47</td>
</tr>
<tr>
<td>( 2^7 \times 2^5 )</td>
<td>SSAAAAAS</td>
<td>0.40</td>
</tr>
<tr>
<td>( 2^7 \times 2^6 )</td>
<td>SSAAAAAS</td>
<td>0.29</td>
</tr>
<tr>
<td>( 2^7 \times 2^7 )</td>
<td>SSAAAAAS</td>
<td>0.23</td>
</tr>
<tr>
<td>( 2^8 \times 2^0 )</td>
<td>AAAAAAAA</td>
<td>0.75</td>
</tr>
<tr>
<td>( 2^8 \times 2^1 )</td>
<td>AAAAAAAS</td>
<td>0.50</td>
</tr>
<tr>
<td>( 2^8 \times 2^2 )</td>
<td>SAAAAAAS</td>
<td>0.50</td>
</tr>
<tr>
<td>( 2^8 \times 2^3 )</td>
<td>SSAAAAAAS</td>
<td>0.50</td>
</tr>
<tr>
<td>( 2^8 \times 2^4 )</td>
<td>SSAAAAAAS</td>
<td>0.49</td>
</tr>
<tr>
<td>( 2^8 \times 2^5 )</td>
<td>SSAAAAAAS</td>
<td>0.46</td>
</tr>
<tr>
<td>( 2^8 \times 2^6 )</td>
<td>SSAAAAAAS</td>
<td>0.39</td>
</tr>
<tr>
<td>( 2^8 \times 2^7 )</td>
<td>SSAAAAAAS</td>
<td>0.28</td>
</tr>
<tr>
<td>( 2^8 \times 2^8 )</td>
<td>SSAAAAAAS</td>
<td>0.22</td>
</tr>
</tbody>
</table>

*Figure 4.5.5. Continued.*
<table>
<thead>
<tr>
<th>Network Shape</th>
<th>$t_{select}=0$</th>
<th>$t_{select}=1$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Arrangement</td>
<td>NTP</td>
</tr>
<tr>
<td>$2^1 \times 2^0$</td>
<td>A</td>
<td>1.00</td>
</tr>
<tr>
<td>$2^1 \times 2^1$</td>
<td>S</td>
<td>0.75</td>
</tr>
<tr>
<td>$2^2 \times 2^0$</td>
<td>AA</td>
<td>1.00</td>
</tr>
<tr>
<td>$2^2 \times 2^1$</td>
<td>SA</td>
<td>0.75</td>
</tr>
<tr>
<td>$2^2 \times 2^2$</td>
<td>SS</td>
<td>0.61</td>
</tr>
<tr>
<td>$2^3 \times 2^0$</td>
<td>AAA</td>
<td>1.00</td>
</tr>
<tr>
<td>$2^3 \times 2^1$</td>
<td>SAA</td>
<td>0.75</td>
</tr>
<tr>
<td>$2^3 \times 2^2$</td>
<td>SAS</td>
<td>0.65</td>
</tr>
<tr>
<td>$2^3 \times 2^3$</td>
<td>SSS</td>
<td>0.53</td>
</tr>
<tr>
<td>$2^4 \times 2^0$</td>
<td>AAAA</td>
<td>1.00</td>
</tr>
<tr>
<td>$2^4 \times 2^1$</td>
<td>SAAA</td>
<td>0.75</td>
</tr>
<tr>
<td>$2^4 \times 2^2$</td>
<td>SAAS</td>
<td>0.67</td>
</tr>
<tr>
<td>$2^4 \times 2^3$</td>
<td>SASS</td>
<td>0.57</td>
</tr>
<tr>
<td>$2^4 \times 2^4$</td>
<td>SSSS</td>
<td>0.48</td>
</tr>
<tr>
<td>$2^5 \times 2^0$</td>
<td>AAAAA</td>
<td>1.00</td>
</tr>
<tr>
<td>$2^5 \times 2^1$</td>
<td>SAAAA</td>
<td>0.75</td>
</tr>
<tr>
<td>$2^5 \times 2^2$</td>
<td>SAAAS</td>
<td>0.69</td>
</tr>
<tr>
<td>$2^5 \times 2^3$</td>
<td>SAAS</td>
<td>0.60</td>
</tr>
<tr>
<td>$2^5 \times 2^4$</td>
<td>SSASS</td>
<td>0.53</td>
</tr>
<tr>
<td>$2^5 \times 2^5$</td>
<td>SSSSS</td>
<td>0.45</td>
</tr>
<tr>
<td>$2^6 \times 2^0$</td>
<td>AAAAAA</td>
<td>1.00</td>
</tr>
<tr>
<td>$2^6 \times 2^1$</td>
<td>SAAAAA</td>
<td>0.75</td>
</tr>
<tr>
<td>$2^6 \times 2^2$</td>
<td>SAAAAAS</td>
<td>0.70</td>
</tr>
<tr>
<td>$2^6 \times 2^3$</td>
<td>SAAS</td>
<td>0.62</td>
</tr>
<tr>
<td>$2^6 \times 2^4$</td>
<td>SSAASS</td>
<td>0.55</td>
</tr>
<tr>
<td>$2^6 \times 2^5$</td>
<td>SSASSS</td>
<td>0.49</td>
</tr>
<tr>
<td>$2^6 \times 2^6$</td>
<td>SSSSSS</td>
<td>0.43</td>
</tr>
</tbody>
</table>

**Figure 4.5.6.** Optimal (buffered) arbitration networks (model (ii), single buffer between stages).
<table>
<thead>
<tr>
<th>Network Shape</th>
<th>t_select=0</th>
<th>t_select=1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Arrangement</td>
<td>NTP</td>
</tr>
<tr>
<td>$2^7 \times 2^0$</td>
<td>AAAAAAA</td>
<td>1.00</td>
</tr>
<tr>
<td>$2^7 \times 2^1$</td>
<td>SAAAAAA</td>
<td>0.75</td>
</tr>
<tr>
<td>$2^7 \times 2^2$</td>
<td>SAAAAAS</td>
<td>0.71</td>
</tr>
<tr>
<td>$2^7 \times 2^3$</td>
<td>SAAASAS</td>
<td>0.64</td>
</tr>
<tr>
<td>$2^7 \times 2^4$</td>
<td>SAAASS</td>
<td>0.57</td>
</tr>
<tr>
<td>$2^7 \times 2^5$</td>
<td>SSASSS</td>
<td>0.52</td>
</tr>
<tr>
<td>$2^7 \times 2^6$</td>
<td>SSASSS</td>
<td>0.47</td>
</tr>
<tr>
<td>$2^7 \times 2^7$</td>
<td>SSASSSS</td>
<td>0.42</td>
</tr>
</tbody>
</table>

$2^8 \times 2^0$ | AAAAAAAA | 1.00 | AAAAAAAA | 1.00 |
| $2^8 \times 2^1$ | SAAAAAAA | 0.75 | AAAAAAAS | 1.00 |
| $2^8 \times 2^2$ | SAAAAAAS | 0.71 | SAAAAAAS | 0.75 |
| $2^8 \times 2^3$ | SAAAAASAS | 0.65 | SAAAAASS | 0.71 |
| $2^8 \times 2^4$ | SASASAS | 0.59 | SASASS | 0.64 |
| $2^8 \times 2^5$ | SASASS | 0.54 | SASASSS | 0.57 |
| $2^8 \times 2^6$ | SSASSSS | 0.49 | SSASSSS | 0.52 |
| $2^8 \times 2^7$ | SSASSSS | 0.45 | SSASSSS | 0.47 |
| $2^8 \times 2^8$ | SSSSSSS | 0.41 | SSSSSSS | 0.41 |

Figure 4.5.6. Continued.
for each possible arrangement of stages for a given number of network input and output links and then picking the one with the highest NTP.

We now briefly discuss the performance of buffered augmented delta networks. In the approximate analysis of the previous section, the number of stages, and not the number of links in the network, determines NTP and NTT. This indicates that a \((b^n \times b^n)\) augmented delta network with \((n + 1)\) stages, has approximately the same NTP and NTT as a \((b^{n+1} \times b^{n+1})\) delta network. This was verified by simulation experiments. It was found that NTP and NTT of a \((2^n \times 2^n)\) augmented delta network, with \((n + 1)\) stages, was indistinguishable from that of \((2^{n+1} \times 2^{n+1})\) delta network. It was observed in the previous section that, for large networks, NTP decreases very slowly, and NTT increases very slowly, with increase in \(n\). Thus, the normal (fault free) performance of an augmented delta network is only slightly lower than that of the delta network of the same size. The topology that results in the event of a worst case single fault in the network is shown in fig. 3.4.3. TP of a \((b^n \times b^n)\) augmented delta network with \((n + 1)\) stages then reduces to that of a \(b^{n-1} \times b^{n-1}\) delta network. Thus, the throughput under the worst case single fault is a little better than half the fault free performance.
4.6. **Chapter Summary**

In this chapter we have studied the performance of buffered delta networks and compared this to the performance of unbuffered networks. The main results of this chapter can be summarized as follows.

(i) Simulation and approximate analysis suggest that there is no significant difference in the performance of topologically different buffered delta networks of the same size.

(ii) Using a binomial approximation in the analysis of buffered delta networks produces good performance estimates (within 5% of simulation results) for \((2^n \times 2^n)\) delta networks with up to 5 stages. However, for networks with many stages the estimated NTP using this approximation overestimates the simulation and second order estimates. This approximation can be generalized to \((b^n \times b^n)\) delta networks with integer \(b\) greater than two, where it can be used to identify trends in performance.

(iii) A second order approximate estimate produces performance estimates within 5% of simulation results for \((2^n \times 2^n)\) delta networks with up to eight stages. (The approximation is probably good for even larger nets but the biggest network simulated had eight stages.) The second order estimate is not feasible for \((b^n \times b^n)\) delta networks with \(b\) greater than two because too many states result, making it intractable.
(iv) For fixed size basic switches and a single buffer between network stages, NTT at maximum load, referred to as the normalized bandwidth (NBW), decreases monotonically with an increase in the network size. However, the rate of decrease in NBW falls rapidly and NBW is almost constant for large networks. For this case, NTT at the maximum load first increases with network size, reaches a maximum and then starts decreasing.

(v) Comparing the performance of buffered delta networks with unbuffered networks of the same size shows that the pipelining effect in buffered nets produces approximately an n-fold improvement in the bandwidth. In model (i) for \( t_{\text{pass}} < t_{\text{select}} \), the improvement is significantly better than n-fold and this holds for all values of \( t_{\text{pass}} \) and \( t_{\text{select}} \) in model (ii). Buffering also produces an improvement in TT. NTT for the unbuffered delta networks increases almost linearly with the number of stages in the networks while, with buffering, NTT reaches a peak and then starts decreasing with an increase in the number of stages. For large networks, buffered delta networks perform better than unbuffered crossbar switches.

(vi) Increasing the number of buffers between the stages of the network leads to a trade-off between the bandwidth and TT. However, the bandwidth quickly saturates to a constant while TT at the maximum load increases almost linearly with the
number of buffers. Thus, for most applications, the number of buffers between stages should be restricted to one or two.

(vii) For model (i), the performance is sensitive to the values of t\_select and t\_pass in an implementation. For a single buffer between stages and for large networks, the bandwidth for t\_select = 0 is almost half that for the case of t\_pass = 0 (for constant t\_delay). For a large number of buffers between stages and large networks, the bandwidth for t\_select = 0 is 0.75 that for t\_pass = 0. The bandwidth for other values of t\_select and t\_pass is between these extremes. The best TT is also obtained for t\_pass = 0.

(viii) If t\_select is less than t\_pass in an implementation, then the performance can be improved considerably by using model (ii) for the operation of a basic switch. For this model, the performance is relatively insensitive to the value of t\_select and t\_pass, particularly for large networks.

(ix) The (b x b) basic switches in a \((b^n x b^n)\) delta network with \(b = 2^k\) can be replaced by \((2^k x 2^k)\) delta networks to obtain a \((2^{n\cdot k} x 2^{n\cdot k})\) delta network composed of \((2 x 2)\) basic switches. Thus the \((b x b)\) switches can be constructed internally as smaller buffered delta networks. This leads to better pipelining and good performance. Further, the analysis for \((2^n x 2^n)\) delta networks can be used to estimate the performance of such networks.
For regular pruned delta networks with internal buffers, in which all switches in the network have the same speed, extremes in performance are obtained for extremes in the number of switches in the network. If the stages in a pruned delta network are matched so that each stage can support the same packet flow rate, then networks that have the best bandwidth have non-intuitive structures and have an intermediate number of switches between those of the extreme choices.

For augmented delta networks, the normal (fault free) performance is slightly worse than that of the delta network of the same size. The performance under a worst case single fault is a little better than one half that under normal operation.
CHAPTER 5

Feedback Networks

5.1. Introduction

The \((b^n \times b^n)\) delta networks that have been studied in the preceding chapters consist of \(n\) stages, \(S_0, \ldots, S_{n-1}\), interconnected by \((n-1)\) permutations, \(P_0, \ldots, P_{n-2}\), as shown in fig. 5.1.1(a). We observed in chapter 2 that it is possible to construct delta networks in which the permutations \(P_0, \ldots, P_{n-2}\) are all identical. The (generalized) perfect shuffle [Stone71a] is such a permutation (and is defined later in the chapter). If \(P_0, \ldots, P_{n-2}\) are the same then \(n\) stages are not necessary. Instead, feedback paths from the last stage of the network, say stage \(S_k\) for \(0 \leq k < n\), to stage \(S_0\) can be added as shown in fig. 5.1.1(b) for \((2^n \times 2^n)\) networks.

In this chapter we will consider networks with \(2^n\) network input and output links, composed of \((2 \times 2)\) switches, with the perfect shuffle permutation between stages, \(k\) stages for \(1 \leq k < n\), and with feedback paths from stage \(S_{k-1}\) to stage \(S_0\) (fig. 5.1.1(b)). These networks will be denoted by \(SEN(2,n,k)\) (short for Shuffle Exchange Network) and are easily generalized to \(SEN(b,n,k)\) which are composed of \((b \times b)\) switches. The single stage \(SEN(2,n,1)\) is similar to the single stage shuffle
(a) A \((b^n \times b^n)\) delta network.

(b) Adding feedback paths.

Figure 5.1.1 Constructing an SEN(2,n,k).
exchange network studied in [Stone7la; Lang76a; Lang76b]. The SEN(2,n,n), without the feedback paths, is similar to the OMEGA network [Lawrie75a]. Both these networks have been studied for their permutation capability. We will study these networks in a packet communication environment similar to that used in the preceding chapters. Some related concurrent research on the SEN(2,n,1) in a packet communication environment is reported in [Lawrie80a]. Part of this study has been reported in [Dias80a].

The motivation for adding these feedback paths is as follows. First, reducing the number of stages in the networks leads to a reduction in the network cost. Of course, there is a concomitant reduction in the performance. In a \((b^n \times b^n)\) network with \(k < n\) stages, packets need to pass through the network several times to get to their destinations. Thus, part of the network bandwidth is used in recycling packets and the bandwidth as seen at the network input and output ports is reduced. However, the bandwidth of a delta network may be more than that required in an application. In the feedback networks, a range of network bandwidth can be obtained by varying the number of stages in the network \((k)\) from 1 to \(n\). A second reason for considering feedback networks is that in these networks there are several paths from input to output links. This leads to fault tolerant capability.

In the asynchronous packet communication environment that we consider, the networks can deadlock. By this we mean that some packets can get permanently blocked in the network. We specify conditions for the
occurrence of a deadlock. These conditions are used in devising schemes for deadlock detection and recovery. Bounds on the performance of the networks are derived and simulation results for the performance of these networks are presented. These simulations vary the number of input sources, stages, buffer lengths between stages, certain other model delay parameters, and the rate at which packets arrive at the network input links. Simulation results indicate that a range of maximum performance can be obtained by varying the number of stages in the network. The $\text{SEN}(2,n,n)$ has the same performance as the same size delta network. However, when networks with a small number of stages are operated at very high packet input rates, deadlocks occur very often and the performance deteriorates rapidly.

Efficient operation of the networks require some control of the flow of packets into the network. One possibility that we examine is to control the number of packets from each source that are simultaneously in the network. It is then possible to guarantee the avoidance of deadlocks by increasing the size of buffers between the stages of the network. However, the number of buffers required to eliminate the occurrence of deadlocks is very large. Simulation results indicate that relatively few buffers are required to make deadlocks occur very infrequently.
5.2. The Basic Model

In this section we define a basic model for the operation of the networks. This model does not take into account the possibility of packets getting deadlocked in the network. In the next section we show that deadlocks can occur in this model and propose refinements to circumvent their occurrence.

Network Topology

We first define the shuffle permutation.

Definition. For an n bit binary number $b_{n-1} \ldots b_0$,

$\text{SHUFFLE}_n(b_{n-1} \ldots b_0) = b_{n-2} \ldots b_0 b_{n-1}$.

The topology of the SEN(2,n,k) is similar to that of the $(2^n \times 2^n)$ delta network with a shuffle permutation between stages as shown in fig. 5.1.1. Note however that there are four input links to switches in stage $S_0$ and four output links for switches in stage $S_{k-1}$. Further, we need to introduce addresses for internal links of the network to specify the shuffle interconnection. The network topology can be specified as follows.

A SEN(2,n,k) (fig. 5.1.1(b)) has $2^n$ network input and $2^n$ network output links, each labelled uniquely by n-bit numbers. It is composed of k stages, $S_0 \ldots S_{k-1}$. Each stage has $2^{n-1}$ switches labelled uniquely by $(n-1)$ bit binary numbers. For $k > 1$, switches in stage $S_0$
have four input links and two output links, while switches in stage $S_{k-1}$ have two input links and four output links. Switches in all other stages have two input links and two output links. For $k = 1$, all switches have four input links and four output links. Each switch input and output link has two single bit labels called the number label and the type label. These labels are shown in fig. 5.1.1(b) as (number label),(type label). Links of type 0 are internal to the network while type 1 links correspond to network input and output links. Each switch has two input (output) links with type label of 0 and these have unique number labels of 0 and 1. The switches in stage $S_0$ have two input links of type 1 with different number labels while stage $S_{k-1}$ switches have two output links of type 1 with different number labels. A link address is obtained by concatenating the switch number and the link number label with the latter as the least significant bit. The stages are interconnected as follows. A switch output link with address $m = m_{n-1} \cdots m_0$ and link type 0 in stage $j$ is connected to input link with address $\text{SHUFFLE}_n(m)$ and type 0 in stage $(j+1) \mod k$. An example of a SEN(2,3,1) is shown in fig. 5.2.1.

Packet routing

We now describe how packets are routed through the networks. The technique is a simple extension of the bit controlled routing used in delta networks.
Figure 5.2.1  An SEN(2,3,1).
A packet arriving at a network input link is assumed to contain both the data to be transferred and the address of the network output link to which the data is to be passed. Consider a SEN(2,n,k) and a packet incident on network input link with label $m_i$ and directed to network output link labelled $m_0$. Let $m_0 = d_{n-1} \cdots d_0$. A count is maintained of the number of network stages that the packet has yet to pass through. Let this count be denoted by stage_count. Each packet passes through $\text{max\_count} = \binom{|R|}{|K|}$ network stages. Stage_count is initialized to $\text{max\_count}$ and is decremented by unity after a packet passes through a stage. If stage_count is greater than $n$, the packet is routed arbitrarily to any output link of the crossbar switch that it is incident on. If $1 < \text{stage\_count} < n$, the packet is directed to the switch output link with type label 0 and link number of $d_{(\text{stage\_count}-1)}$ (i.e., the packet remains in the net and its routing is determined by a particular bit of the destination label). If stage_count = 1, the packet is directed to the switch output link with type label 1 and link number of $d_0$ (i.e., the packet is removed from the network after it has passed through $\text{max\_count}$ stages).

The correctness of this routing algorithm follows by an easy induction on the number of stages that the packet has passed through and the definition of the SHUFFLE permutation.

The preceding routing algorithm does not route packets through the network along the shortest path between network input and network output.
links. Instead, routing efficiency is sacrificed for routing simplicity. The algorithm can be easily modified to direct a packet along the shortest path if some preprocessing is done to determine this path. The algorithm is also easily modified to route packets around faulty switches. If a faulty switch is encountered by a packet, the packet can be sent to the other link at the switch and the stage_count appropriately reset. We will not explore these possibilities further in this thesis.

The Environment, Performance Criteria and Switch Operation

We make the same assumptions about the environment and use the same performance criteria as those specified in section 4.2. Model (i) of section 4.3 is used to model the operation of the basic switches in the network.

5.3. Deadlocks

We first define what we mean by deadlocks in this context.

Definition. A packet $p$ incident at a switch in an $SEN(2,n,k)$ at time $t$ with marking $M$ of the Petri net corresponding to the SEN (see appendix 1) is said to be blocked if

(i) it corresponds to a token in an output request flag (fig. 4.3.1) (i.e., the output link to which it must be passed has been selected) and the successor buffer empty flag does
not have a token (fig. 4.3.1) (i.e., the successor input buffer is full),

or, (ii) it corresponds to a token in an input queue and the packet at the head of the queue is blocked (i.e., it is in a first in first out queue in which the packet at the head of the queue is blocked).

**Definition.** A packet incident at a switch in an SEN(2,n,k) at time $t_1$ is said to be **deadlocked** if it is permanently blocked (i.e., it is blocked at time $t_1$ and remains blocked at all times $t > t_1$).

We will present a simple sufficient condition that can be used to detect all deadlocks eventually. By this we mean that the condition does not detect a deadlock as soon as it occurs but will detect it at some later time. We have been unable to obtain a necessary and sufficient condition for the occurrence of a deadlock. Such a condition is hard to find because of subtle timing considerations in the timed Petri net. It is possible to determine necessary and sufficient conditions for the occurrence of a deadlock if the Petri net model is interpreted as an untimed Petri net. However, these conditions are sufficient but not necessary in the timed Petri net. Further, they are difficult to use in a feasible deadlock detection scheme. We need the following definitions.

**Definition.** A path of blocked queues at time $t$ and marking $M$ is a sequence of queues $q_1 \ldots q_m$ at links $l_1 \ldots l_m$ of switches $s_1 \ldots s_m$
respectively such that there are blocked packets in each of these queues and the packet \( p_j \) that is at the head of queue \( q_j \) for \( 1 \leq j < m \) is directed to link \( l_{j+1} \).

**Definition.** A cycle of blocked queues at time \( t \) and marking \( M \) is a path of blocked queues \( q_1 q_2 \ldots q_m q_1 \) where \( q_2, \ldots, q_m \neq q_1 \). Here \( m \) is referred to as the length of the cycle.

**Lemma 5.3.1.** Sufficient condition for a packet to be deadlocked at time \( t \) and marking \( M \).

A packet \( p \) incident in a queue of a SEN(2,\( n, k \)) at time \( t \) and marking \( M \) is deadlocked if

\[(i) \quad \text{it is in a cycle of blocked queues}
\]

or \( (ii) \) it is in a path of blocked queues \( q_1 \ldots q_m \) and \( q_m \) is in a cycle of blocked queues.

**Proof.** Trivial.

\[\square\]

**Theorem 5.3.1.** Suppose that a packet \( p \) in a queue of a SEN(2,\( n, k \)) is deadlocked at time \( t \) and marking \( M \). Then there exists a time \( t' \geq t \) with marking \( M' \) at which \( p \) satisfies the sufficient condition of lemma 5.3.1.
Proof. Consider the link $l_1$ to which the deadlocked packet $p$ is directed. From the definition of a blocked packet, the queue at link $l_1$ must be full and the packet $p_1$ at the head of this queue must either be deadlocked at time $t$ and marking $M$ or there must exist a time $t_1 \geq t$ with marking $M_1$ at which packet $p_1$ gets deadlocked in the same queue. This follows easily by contradiction. This reasoning can be repeated for packet $p_1$ and so on. The theorem then follows since the number of links in the net is finite.

\[\square\]

Schemes for deadlock detection.

We now propose some simple schemes for deadlock detection. These schemes involve sending test packets, when a data packet is blocked, in an attempt to determine a cycle of blocked queues. The naive idea that first comes to mind is the following.

(i) Construct test packets containing the address of the switch from which they originate.

(ii) Send a test packet to a successor switch when a data packet at the head of a queue gets blocked.
(iii) Forward test packets along a path of blocked queues.

(iv) Discard test packets that reach an unblocked queue.

(v) Flag a deadlock if a test packet returns to its source link.

The main problem with this scheme is that the test packets can themselves deadlock. To eliminate the possibility of test-packet deadlock, the schemes proposed will have test packets consisting of a single bit. A count of test bits received will be maintained and knowledge of the maximum loop length in the network will be used to detect the occurrence of a loop of blocked packets.

In the following, test bits will be sent along a link only if a packet that is to be passed along that link is blocked. We assume that a suitable protocol exists to prevent simultaneous passage of a test and a data packet along a link. Let $L = k \cdot 2^n$ denote the number of links of type 0 (i.e., internal links). Thus $L$ is the maximum length loop in an SEN$(2,n,k)$. Test bits are advanced along a blocked path. When a deadlock occurs, test bits thus forwarded circulate indefinitely. Knowledge of $L$ in a network is used to determine when this occurs.

**Deadlock detection scheme 1 (DDS1)**

In this scheme a count, denoted by $c_{\text{sum}}$, is maintained at each switch output link of type 0. The value of this count is the number of test packets generated and forwarded through this output link since a data packet was blocked at this output. The count is modified as shown.
in fig. 5.3.1. This figure is a Petri net model for the operation at a switch output link. The transitions in the figure represent events in the system. The places marked 'blocked' and 'unblocked' represent the state of the output link. The operation is essentially as follows.

(i) When the system is initialized, c_sum is set to zero and the output is set to the unblocked state (as indicated in the Petri net).

(ii) A test bit is generated at a switch output link iff a data packet gets blocked at that output link (i.e., either a data packet sets the output request flag and the corresponding output buffer is full or a data packet is waiting for passage to an output link and the buffer gets full). When a test bit is generated, c_sum is also incremented. In the Petri net of fig. 5.3.1, a data packet becoming unblocked is shown as an external event that results in the desired action.

(iii) Test bits that arrive at a full queue are directed to the output link to which the packet at the head of the queue is directed. (The test bit is held until the t_select phase for this packet is over.) If this data packet is blocked, the test bit is forwarded to the successor switch input link (which must be full) and c_sum is incremented.
Figure 5.3.1  Petri net model for DDSI.
(iv) If a blocked data packet becomes unblocked, $c_{\text{sum}}$ is set to zero.

(v) When $c_{\text{sum}}$ exceeds or equals $L$, a deadlock is flagged.

Essentially, this scheme generates test bits when data packets become blocked, forwards test bits along a blocked path, and keeps a count of the number of test bits sent from an output link since a data packet at that output link becomes blocked. We now show that this scheme does detect a cycle of blocked queues and we bound the time that it takes to do so after the cycle occurs. We need the following lemmas.

**Lemma 5.3.2.** Suppose that there is a path of blocked queues $q_1 \cdots q_m$ at time $t$ in an SEN($2,n,k$) operating under the assumptions of section 5.2. Then, if queue $q_m$ remains blocked at all times in the interval $[t,t']$, with $t'\geq t$, then so do queues $q_1 \cdots q_{m-1}$.

**Proof.** Follows easily by an induction on $m$.

In the following lemma we need to identify the link from which a test bit originates. This will enable us to bound the number of packets received by a switch. Thus, in the following lemma we will assume that it is possible to identify the source link of a test bit (say by an associated tag). This is strictly for the purpose of proving the
correctness of DDS1 and is not necessary in an implementation.

**Lemma 5.3.3.** Suppose that a test bit is generated at time $t$ at a switch $S$, output link $L$ by a data packet, located at an input link of $S$ with queue $q$, becoming blocked. Suppose that this test bit is received at time $t' \geq t$ at another switch $S'$, output link $L'$ from an input link of $S'$ with a blocked queue $q'$. Then

(a) there is a path of blocked queues $q \ldots q'$ at time $t'$ and

(b) queue $q$ remains blocked in the time interval $[t, t']$.

**Proof.** By an induction on the number of switches, $x$, through which the test bit passes in going from switch $S$ to switch $S'$.

**Basis.** For $x = 1$, switch $S'$ is the immediate successor of switch $S$.

Thus, $q$ is the same as $q'$ and the lemma follows immediately from lemma 5.3.2.

**Induction step.** Suppose that the lemma is true for $x = k$. Since test bits are only forwarded at a blocked output, statement (a) is true for $x = k+1$. Statement (b) is true for $x = k+1$ from lemma 5.3.2.

**Lemma 5.3.4.** Suppose that no cycle of blocked queues occurs in an SEN($2, n, k$) using DDS1. Then the maximum count $c_{\text{sum}}$ that can be accumulated at a switch output link is $L-1$.

**Proof.** If no packet is blocked at that output link then $c_{\text{sum}}$ must be zero from (iv) of DDS1 and the corresponding Petri net. Suppose that
only one packet is blocked at this output link (i.e., a data packet from the other input link is not blocked at this output). Then exactly one test bit is generated at this link. Further, from lemma 5.3.3 at most (L-2) test bits are received at this link. (If L-1 or more bits are received then L queues are blocked and there must be a cycle of blocked queues.) Thus the maximum accumulated count is L-1. Suppose that two packets (from different input links) are blocked at the link in question. Then exactly two test bits are generated at that output link. Further, the other output link of the switch cannot be blocked (since no packet is directed to it). Then at most L-3 test bits can be received at that output by the same reasoning as earlier, and since the other output link of the switch cannot be blocked.

Lemma 5.3.5. If a cycle of blocked queues occurs in an SEN(2,n,k) using DDS1 then, the number of test bits received at switch links in the cycle exceeds L.

Proof. If this occurs then test bits circulate indefinitely and the lemma follows.

Now suppose that it takes time $t_{test}$ to pass a test bit from one blocked switch output link to an output link of the successor switch. The following bound on the time to detect a cycle of blocked packets can be established.
Theorem 2. DDS1 detects a cycle of blocked packets at most in time $t_{test} \cdot m \cdot L$ after it occurs, where $m$ is the length of the cycle.

Proof. Follows directly from lemmas 5.3.4 and 5 and since there is at least one test bit that circulates indefinitely in the blocked cycle.

This bound on the time to detect a cycle of blocked queues is tight in that execution sequences can be constructed in which only one test bit circulates in the blocked cycle. The reason for this is as follows. In DDS1, when a data packet gets unblocked the count $c_{sum}$ at that output is set to zero. However, the same data packet may subsequently get blocked again at the same output link. If this happens, the information of the number of test bits received is still valid but is lost. However, if the data packet is advanced at that output link then, the input queue from which it is moved is no longer full and the test bits received are to be discarded. The performance of the detection scheme can be improved by modifying DDS1 as follows.

Deadlock Detection Scheme 2 (DDS2)

In this scheme three counts, denoted by $c_{sum}$, $c_0$ and $c_1$, are maintained at each switch output link of type 0. The counts are modified as shown in the Petri net model of fig. 5.3.2. The operation is essentially as follows.
Figure 5.3.2 Petri net model for DDS2.
(i) When the system is initialized, \( c_{\text{sum}} \), \( c_0 \), and \( c_1 \) are set to zero and the output is set to the unblocked state.

(ii) If a data packet from input link \( i \) of the switch gets blocked at that output link, \( (c_1 + 1) \) test bits are generated and \( c_{\text{sum}} \) incremented by \( (c_1 + 1) \).

(iii) If a test bit arrives at a full queue at input \( i \) of a switch \( S \), it is directed to the output link to which the packet at the head of the queue is directed. (The test bit is held until the \( t_{\text{select}} \) phase for this packet is over.) If this data packet is blocked, the test bit is forwarded to the successor switch input queue (which must be full) and \( c_1 \) and \( c_{\text{sum}} \), incremented.

(iv) If a blocked data packet becomes unblocked, \( c_{\text{sum}} \) is set to zero.

(v) If input queue \( i \) becomes not full (from being full) set \( c_1 \) to zero.

(vi) When \( c_{\text{sum}} \) exceeds or equals \( L \), a deadlock is flagged.

\( \square \)

Theorem 5.3.3. DDS2 detects a cycle of blocked packets at most in time \( t_{\text{test}} \cdot L \) after it occurs.
Proof. Similar to theorem 5.3.2 except that there are m test bits that circulate in a cycle of blocked queues.

This bound on the deadlock detection time will be used in the next section to lower bound the performance of the networks.

Deadlock Recovery.

After a cycle of blocked packets is detected, the deadlock condition can be recovered from by advancing packets in the blocked cycle. This prevents the condition of theorem 5.3.1 from causing a deadlock. This can be achieved in an implementation by reserving a special buffer for deadlock recovery and advancing packets in the detected loop of blocked packets. The advancing can occur in parallel and thus takes a constant time which we will denote by $t_{\text{recovery}}$.

Theorem 5.3.4. Deadlocks caused by a cycle of blocked packets can be recovered from by advancing packets in the cycle (by one switch).

Proof. Similar to the proof of theorem 5.3.1 and using the fact that in an SEN(2,n,k) packets only pass through a finite number of stages.

In DDS1 and DDS2 it is necessary to reset all the counts at output links in the detected cycle of blocked packets when recovering from the deadlock.
5.4. Performance

In this section we present simulation results for the performance of these networks. Simulations that vary the number of network input and output links \( (2^n) \), the number of stages \( (k) \), buffer lengths between stages, \( t\text{-select}, t\text{-pass}, t\text{-test} \) and \( t\text{-recovery} \) have been performed and some typical results are reported here. In these simulations the upper bound for deadlock detection time of theorem 5.3.3 is used. Thus, when a cycle of blocked packets occurs in the simulation model, it is assumed that deadlock detection takes time \( (t\text{-test} \cdot L) \), followed by a \( t\text{-recovery} \) period to advance packets in the blocked cycle by one place. Since this is the upper bound for the time required for deadlock detection and recovery, the simulation results are lower bounds on performance. It is also assumed that passing test bits does not affect the normal (deadlock free) performance of the networks. Since test bits are only passed along a path of blocked data packets, this is a reasonable assumption. We also consider controlling the flow of packets into the network as a means of improving performance.

The performance criteria used are the same as in chapter 4. Denote \( (t\text{-select} + t\text{-pass}) \) by \( t\text{-delay} \). The minimum delay that a packet encounters in an SEN\((2,n,k)\) is \( \min_{tt} = k \cdot \frac{|n|}{|k|} \cdot t\text{-delay} \). This follows directly from the packet routing algorithm. An upper bound on throughput at the maximum load is denoted by \( \max tp2 \). It is the throughput that would be obtained if there were no conflicts in the network. With this assumption, the bandwidth of a switch is the only factor that
limits thruput. For \( 1 < k \leq n \), \( \text{maxtp}_2 = \frac{1}{\frac{n!}{k!} \cdot t_{\text{delay}}} \). This is because each packet passes through the network \( \frac{n!}{k!} \) times and internal links in the network must handle all this load. For \( k=1 \) and \( n>1 \), (i.e. for networks with a single stage and larger than a single switch) \( \text{maxtp}_2 = \frac{1}{(n - 1) \cdot t_{\text{delay}}} \). This bound derives from the fact that the feedback paths are the bottleneck for this case. Each packet passes \( (n-1) \) times through the feedback paths and then out of the network.

We also consider the case of an exponential interarrival time between packets after a buffer becomes available at an input link. Let \( \tau \) denotes the average interarrival time. An upper bound on the thruput for this case is denoted by \( \text{maxtp}_1 \) and is obtained using similar assumptions to that for \( \text{maxtp}_2 \). \( \text{Maxtp}_1 \) is bounded by \( \text{maxtp}_2 \) and by the maximum average input rate. Thus, \( \text{maxtp}_1 = \min(\text{maxtp}_2, \frac{1}{\tau}) \).

Fig. 5.4.2 shows how \( \text{maxtp}_2 \) varies with the number of stages for an \( \text{SEN}(2,5,k) \), for \( 1 \leq k \leq 5 \). Fig. 5.4.1 shows the ratio of \( \text{maxtp}_1 \) to \( \text{maxtp}_2 \) with a varying \( \tau \) for the same case. Observe that though \( \text{maxtp}_2 \) is a monotonically increasing function of \( k \), it is not a strictly increasing function of \( k \). This is because the changes in \( \text{maxtp}_2 \) only occur when there is a change in the number of times a packet has to pass through the network. The thruput normalized with respect to \( \text{maxtp}_1 \) and \( \text{maxtp}_2 \) will be denoted by \( \text{NTP}_1 \) and \( \text{NTP}_2 \) respectively.

We now examine some typical simulation results for the variation of
Figure 5.4.1 maxtp1/maxtp2 versus inter-arrival time for an SEN(2,5,k), 1 \leq k \leq 5.

Figure 5.4.2 maxtp2 versus k for an SEN(2,5,k), 1 \leq k \leq 5.
NTP1 and NTP2 with $\tau$ and length of buffers between stages. Fig. 5.4.3 shows the variation of NTP1 with $\tau$ for an SEN(2,5,k). The buffer size is the parameter in this graph. Notice that for a fixed buffer size NTP1 increases with $\tau$. This is to be expected since larger $\tau$ implies lower loads on the network and this leads to fewer conflicts and fewer deadlocks in the network. For large $\tau$, NTP1 approaches unity implying that there are very few conflicts and deadlocks at low loads. However, for small $\tau$, NTP1 is very poor. This figure also indicates that, for a fixed $\tau$, NTP1 increases with an increase in the size of buffers between stages. Again, increasing buffer size at a fixed input arrival rate can be expected to reduce the frequency of conflicts and deadlocks leading to improved performance. Note, however, that the for small $\tau$, the improvement in NTP1 with increase in the buffer size is marginal. A closer look reveals that NTP1 starts increasing rapidly as $\tau$ increases beyond 4. This is precisely the value at which $\frac{1}{\tau}$ equals $\maxtp2$. Thus, if the input arrival rate exceeds or is near the maximum rate that the network can handle, the network clogs up and performance is very poor. This effect is more pronounced for large buffer sizes.

The variation of NTP2 with $\tau$ and buffer size is shown in fig. 5.4.4. For $\frac{1}{\tau}$ greater than $\maxtp2$, $\maxtp1$ is the same as $\maxtp2$ and NTP2 is also the same as NTP1. Thus the comments for NTP1 apply for this region of the graph. Recall that $\maxtp2$ is a bound on the throughput at maximum load and is thus independent of the interarrival time. Observe that the NTP2 first increases with the interarrival time,
Figure 5.4.3 NTP1 versus inter-arrival time for an SEN(2,5,1) with a variable buffer size.
Parameters:  t_select = 0;  t_pass = 1;
            t_test = 0.1;  t_recovery = 1;

Figure 5.4.4 NTP2 versus inter-arrival time for an SEN(2,5,1) with a variable buffer size.
Parameters:  t_select = 0;  t_pass = 1;
            t_test = 0.1;  t_recovery = 1.
reaches a peak and then starts falling. The peak is more pronounced for larger buffer sizes. This means that the thruput behaves in the same fashion. The explanation is that deadlocks occur very often at very high loads and this drastically reduces the performance. As the load on the network decreases, the frequency of deadlock occurrence also falls leading to an improvement in the network thruput. This suggests that some control of the packet arrival rate is needed to obtain good performance by preventing frequent deadlocks.

The variation of NTPl with $\tau$ and $k$ (the number of stages in the network) is shown in fig. 5.4.5. The NTPl increases with $k$ for the same $\tau$. This is because the frequency of deadlock occurrence falls with increase in the number of stages. In fact, the SEN(2,$n$,$n$) never deadlocks since no packets are fed back under normal operation of the network. Figure 5.4.2 shows that maxtp2 increases monotonically with the number of stages in the network. Thus, a range of network bandwidth can be obtained by varying the number of stages in the network.

We concluded earlier that some control of the flow of packets into the network is necessary for good performance at high loads. We now look at a simple method of doing this. Consider restricting the number of packets from each source that can be in the network simultaneously. Let $p$ denote this number of packets. This can be approximated in practice by destination modules acknowledging the arrival of a packet, and an input controller preventing the difference between the number of packets sent and acknowledged from exceeding a certain value.
variable parameter: number of stages \((k)\)

**Figure 5.4.5**  NTPI versus inter-arrival time for an SEN(2,5,k), with \(1 \leq k \leq 5\).

Parameters (fixed):
- Single buffer between stages;
- \(t_{\text{select}} = 0\); \(t_{\text{pass}} = 1\);
- \(t_{\text{test}} = 0.1\); \(t_{\text{recovery}} = 1\).
Fig. 5.4.6 shows the variation of NTP2 and NTT with p for SEN(2,n,1), 1 < n ≤ 8, with τ = 0 and with five buffers between stages. Note that the NTP peaks for a certain value of p and that this value of p decreases with increase in the size of the network. Fig. 5.4.7 shows similar characteristics for ten buffers between stages. Comparison of these figures shows that as the buffer size increases, the value of p at which NTP2 peaks increases with an increase in the number of buffers between stages. The peak value of NTP2 obtained is very good (about 0.65 for an SEN(2,8,1) with ten buffers between stages and p of three). However, these figures show that though the NTP has a peak at p greater than one, the turn around time increases monotonically with p. The reason is that a packet suffers more conflicts as the number of packets in the network increases. This tradeoff between throughput and turn around time needs to be considered for specific applications.

5.5. Chapter Summary

In this chapter we have briefly considered shuffle exchange networks with a varying number of stages in an asynchronous packet communication environment. The principal results of this chapter are as follows.

1. In the asynchronous packet communication environment that we consider, packets can deadlock within the network. A sufficient condition for a deadlock that will eventually detect all deadlocks is the occurrence of a cycle of blocked packets.
Figure 5.4.6  NTP2 and NTT versus the number of packets from each source simultaneously in the network. Parameters: t_select = 0; t_pass = 1; t_test = 0.1; t_recovery = 1; 5 buffers between stages; SEN(2,5,1)
Figure 5.4.7  NTP2 and NTT versus the number of packets from each source simultaneously in the network.
Parameters:  t_select = 0;  t_pass = 1;  t_test = 0.1;  t_recovery = 1;  10 buffers between stages;
SEN(2,5,1).
(ii) Deadlocks can be detected by passing test packets consisting of a single bit along a path of blocked data packets. We present a deadlock detection scheme that takes $O(L)$ time to detect a cycle of blocked packets where $L$ is the maximum length of a loop in the network.

(iii) Deadlock recovery can be effected by advancing packets in a cycle of blocked packets.

(iv) Simulation results indicate that when networks with $k < n$ stages are used, deadlocks occur very frequently at high loads leading to poor performance even for large buffer sizes. Reducing the load on the network leads to fewer deadlocks and better throughput.

(v) For the same packet inter-arrival time $\tau$, increasing the number of buffers between stages leads to better performance. This is more pronounced at intermediate values of $\tau$ (i.e. between very small $\tau$ where frequent deadlock occurrence deteriorates performance, and very large $\tau$ where the network is under-utilized).

(vi) Varying the number of stages leads to a range of maximum throughput.
If the number of packets from any particular source that can be simultaneously in the network is restricted, deadlocks can be avoided by having suitable large buffers. However, the number of buffers at each link, required to prevent a deadlock altogether, is very large for large networks. Deadlocks occur very infrequently and good performance is obtained for relatively small buffer lengths.
CHAPTER 6

Conclusions

In this chapter we summarize the main results of this thesis and suggest areas for further research.

6.1. Summary

In this thesis we have studied a class of multistage interconnection networks, called delta networks, and networks of related topology, in a packet communication environment. These networks can be used to interconnect a large number of modules in a modular computing system. Typical applications are in multiprocessor memory interconnections and parallel processor communication. The mode of operation can be modified to apply to local computer networks. Packet communication is particularly advantageous (as compared to circuit switching) in systems that send short bursty messages and this is a common environment encountered in computing systems. Computer architectures using packet communication as the basis have been proposed and these have the advantages of allowing for modular design, large parallelism, distributed control, clean interfaces, speed independence, and simple proofs of correctness. The networks we consider exhibit these properties.

Special extreme cases of \((N \times N)\) delta networks are the full crossbar switch, which has \(O(N^2)\) gates and \(O(N)\) ideal bandwidth, and the
single bus which has $O(N)$ gates and $O(1)$ bandwidth. The case that we examine in detail is that of networks constructed from fixed size basic switches which results in $O(N \log N)$ gates and $O(N)$ ideal bandwidth. We show that in realistic environments, the performance of delta networks with fixed size basic switches is comparable to that of the full crossbar switch. In fact, the buffered delta networks can perform better than the commonly encountered unbuffered crossbar switches. The class of delta networks is either topologically equivalent to or includes several other networks that have been proposed in the literature and hence the performance analyses that we have presented also apply to these networks.

Delta networks can route packets from input to output links under local control at the basic switches. They can be constructed recursively from smaller delta networks. Further, the basic switches in a delta network can themselves be replaced by delta networks to yield delta networks with smaller basic switches. Bidirectional delta networks exhibit the local routing capability in both direction (i.e. from input to output links and vice versa) and can also be constructed recursively. We have shown that all bidirectional delta networks are topologically equivalent. Delta and bidirectional delta networks have a unique path from each network input link to each network output link which leads to poor reliability. They can be augmented to provide multiple disjoint paths through the network and still retain local control in routing packets and $O(N \log N)$ gate complexity. The original networks have the same number of input and output links. They can be
pruned to obtain networks with differing numbers of input and output links. However, there are many ways of pruning delta networks and we have examined how the different pruned networks perform.

Unbuffered delta networks have no internal buffers within the network. A simple model for network operation can be used to obtain a network characteristic that can then be used to obtain good estimates of performance for other models and environments. Introducing priority in arbitrating between conflicting packets was found to deteriorate the performance. For regular pruned delta networks, the extremes in performance were obtained for extremes in the number of switches in the network. The normal (fault-free) performance of augmented delta networks is a little worse than that of the same size delta network. The performance for a worst case single fault is a little better than one half that for normal operation.

Introducing buffers between the stages of a delta network leads to a better than $n$-fold improvement in the network bandwidth (where $n$ is the number of stages in the network). This is because the routing of packets through the network is split into $n$ sub-routing steps leading to a pipelining effect. For a single buffer between stages there is also an improvement in the average delay that a packet encounters in passing through the network. Increasing the number of buffers between stages leads to a trade-off between the bandwidth and the delay. However, the bandwidth quickly saturates to a constant with further increase in the number of buffers while the delay increases almost linearly with buffer
size. This suggests that, for most applications, the number of buffers between network stages should be limited to one or two. The performance obtained is also sensitive to precisely how the nodes in the network operate. The node operation policy needs to be chosen depending on the values of certain network parameters in an implementation. For pruned delta networks with internal buffers, in which all switches in the network have the same speed, the results are similar to the unbuffered case. If the stages in a pruned delta network are matched so that each stage can support the same packet flow rate, then networks that have the best throughput are non-intuitive and have an intermediate number of switches between that of the extreme choices.

Introducing feedback paths allows the construction of networks with a varying number of stages and a range of performance. However, packets can deadlock in such networks. It is possible to detect and recover from deadlocks by passing test packets when a (data) packet gets blocked in the network. However, at high loads, the deadlocks occur very frequently and this deteriorates performance. Some control of the flow of packets into the network is necessary to obtain good network performance. If the number of packets that can be simultaneously in the network from any particular source is restricted then it is possible to avoid deadlocks altogether by using suitably large buffers. However, the number of buffers required to avoid deadlocks is very large. Deadlocks can be made very infrequent and good performance obtained by using a relatively small number of buffers.
6.2. Future Research

We conclude this thesis by suggesting some areas for further research.

We have not considered the cost to performance ratio in our study. This is because the cost is implementation dependent and cost criteria differ depending on the specific application. For instance, if the entire network is to be built on a single LSI/VLSI chip, the area occupied on the chip may be a reasonable cost criterion. However, if the system is built form basic switches that are available as chips, then the number of chips used may determine cost. Other possible cost criteria are the length of wire required, the number of links or some weighted average of these criteria. A detailed cost/performance study needs to be done depending on the application at hand.

We have briefly considered the fault tolerant capability of these networks. Detailed reliability studies need to be carried out. Recall that in an augmented delta network, stages can be added to improve the number of faults that the network can handle and this determines the performance of the network with faulty switches. However, adding stages to the network increases the cost and reduces the normal (fault free) performance of the network. The trade-off between reliability, cost and performance needs to be studied.

We observed that the basic switches of which the networks are constructed can be busses or crossbars. The performance studies carried
out were for crossbar type basic switches. The techniques presented in this thesis can easily be extended to other switch types. Various other switch types can be considered. For instance, a basic switch can model an ethernet [Metcalf76a] and the network can be a multiple ethernet local computer network.

Some of the advantages of feedback networks are their fault tolerant capability and the fact that all packets need not pass through the same number of stages. The latter property can be exploited in systems with a high locality in interaction (i.e. systems in which some modules communicate more frequently than others). A mode of operation of these networks different from the one we have considered has been proposed in [Lawrie80a; Padua80a]. A performance comparison of these schemes needs to be done.

In this thesis we have isolated the interconnection network and studied its performance. In several systems, the network's performance plays a part in determining the load on the network. For instance, packets passing out of the network may cause new packets to be produced. Thus the delay that the network imposes on packets may determine the load on the network. Studies of the performance of systems with interconnection networks integrated into them need to be done. We believe that the network characteristics exposed in this thesis and the analysis techniques developed can be used in such studies.
REFERENCES


APPENDIX

Elements of Petri Nets

In this appendix we briefly define Petri nets and timed Petri nets. The definitions are similar to those in [Jotwani77a; Ramachandani74a].

A Petri net is a triple $G=(T,P,E)$ where $T$ is a non-empty set of transitions, $P$ is a non-empty set of places, and $E \subseteq (T \times P) \cup (P \times T)$ is a set of edges. Thus, a Petri net is a bipartite graph, with two types of nodes called places and transitions.

A marking is a function $M: P \rightarrow N$, where $N$ is the set of non-negative integers. Thus, a marking associates $M(P)$ tokens with place $P$.

Notation: $x^*$ denotes the set $\{y \mid (x,y) \in E\}$ and $x^t$, the set $\{y \mid (y,x) \in E\}$.

Firing rules: A Petri net progresses through a sequence of markings by the firing of transitions. A transition $t$ is firable for marking $M$ if for all $p$ in $x^t$, $M(p) > 0$. Firing the transition $t$ leads to a marking $M'$ where,

\[
M'(p) = \begin{cases} 
M(p) + 1, & \text{if } p \in x^t - x^t, \\
M(p) - 1, & \text{if } p \in x^t - x^t, \\
M(p), & \text{otherwise.}
\end{cases}
\]
This is denoted by $M \xrightarrow{t} M'$. Informally, a transition is firable when each of its input places has a token. When a transition fires, it removes a token from each of its input places, and puts a token into each of its output places.

Firing sequences: For an initial marking $M_0$, a firing sequence is a sequence $\sigma : \eta \rightarrow T$, $\eta = 0, 1, 2, \ldots$, which defines a sequence of markings

$$M_0 \xrightarrow{\sigma(0)} M_1 \xrightarrow{\sigma(1)} M_2 \rightarrow \ldots$$

where if $\sigma(i) = t$, then transition $t$ is enabled for marking $M_i$ and $M_i \xrightarrow{t} M_{i+1}$. We extend the notation to say that $M_0 \xrightarrow{\sigma(0) \ldots \sigma(n)} M_{n+1}$ or denoting $\sigma(0) \ldots \sigma(n)$ by $\sigma_n$, $M_0 \xrightarrow{\sigma_n} M_{n+1}$.

These definitions suffice for the purposes of thesis.

Timed Petri Nets

A timed Petri net essentially associates a time interval with the firing of a transition. This allows the modelling of systems where timing considerations are important. With this modification, the operation of the network can occur in real (or model) time. The model can be formalized as follows.
A Timed Petri net is a pair $G' = (G, \Omega)$ where $G = (T, P, E)$ is a Petri net and $\Omega$ is a function $\Omega: T \rightarrow \mathbb{R}$ where $\mathbb{R}$ is the set of non-negative real numbers. The number $\tau_i = \Omega(t_i)$ is termed the firing time of transition $t_i$.

The firing rules and firing sequence are much the same as for Petri nets. A firable transition can initiate a firing. When this occurs, a token is removed from each input place of the transition, and the transition is said to be executing. This execution phase lasts for the firing time of the transition, and at the end of this phase, tokens are placed in the output places of the transition. These ideas are easily formalized, as described in [Ramachandani74a].