WASHINGTON UNIVERSITY

SEVER INSTITUTE OF TECHNOLOGY

PERFORMANCE EVALUATION OF ASYNCHRONOUS TRANSFER MODE SWITCHING SYSTEMS

by

EINIR VALDIMARSSON

Prepared under the direction of Professor Jonathan S. Turner

A thesis presented to the Sever Institute of
Washington University in partial fulfillment
of the requirements for the degree of

DOCTOR OF SCIENCE

December, 1994

Saint Louis, Missouri

WASHINGTON UNIVERSITY
SEVER INSTITUTE OF TECHNOLOGY

ABSTRACT

PERFORMANCE EVALUATION OF ASYNCHRONOUS TRANSFER MODE SWITCHING SYSTEMS

by EINIR VALDIMARSSON

ADVISOR: Professor Jonathan S. Turner

December, 1994
Saint Louis, Missouri

The design of switching systems is an important and active research area because it is essential to the success of broadband communication systems. The work presented in this thesis provides models that evaluate the performance of high speed Asynchronous Transfer Mode switching systems. These models help to evaluate the design trade-offs in switching systems needed to support current and future communication needs.

The work has two parts. The first part provides new analytical models for buffered switching networks. Specifically, models for copy networks with various buffer schemes and a model for shared buffer Benes network with non-uniform traffic are presented. Our results show that the models are accurate for single stage networks, and close approximations for multistage networks.

The second part of the work focuses on the development and use of simulation models of switching systems. Such simulation models are necessary, due to the limitations of analytical models, and can provide valuable information on many complex switching systems. To make evaluation and design of such systems easier and to gain a better understanding we have designed and implemented an efficient tool for evaluating switching systems. The tool is general enough to allow performance evaluation of a variety of different switching architectures. The novel aspects of the tool include easy network construction and interaction, and visualization and animation of network state and statistics.

We have used the models provided by the simulation tool to compare various switching systems in terms of cell loss probability, cell delay, and complexity required to obtain certain performance requirements. Finally, we have used the tool to examine the transient behavior of shared buffer Benes systems and we have obtained new insights about the congestion conditions in the network.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

PERFORMANCE EVALUATION OF ASYNCHRONOUS TRANSFER MODE SWITCHING SYSTEMS

1. INTRODUCTION

Past centuries have each been dominated by some technology. During this century, the key technology has been concerned with information gathering, processing and distribution. Among other developments, we have seen the installation of worldwide telephone networks, the invention of radio and television, the launching of communication satellites, and the birth and unprecedented growth of the computer industry.

As we move toward the end of this century, the areas of telephony, television and computers are rapidly converging, and the differences between collecting, transporting, storing and processing information are becoming unimportant. As our ability to gather, process and distribute information grows, the demand for even more sophisticated tools increases even faster and high speed communication infrastructures become necessary.

The concept of a communication network arises naturally when large numbers of users have the desire to communicate with one another. One would not want to directly and physically connect each pair of users since many of them might communicate only infrequently and every time a new pair of users desires to communicate a new direct connection would be required. The cost associated with communication is reduced with the use of a network. A network consists essentially of switches, or nodes, and user terminals, interconnected by transmission links. It is the role of the network's switching systems to provide connections between users on an as needed basis.

1.1. Motivation and Objective

Effective design of switching systems is critical to the success of communication systems. Poor design may result in problems such as excessive delays in delivering data and inability to satisfy quality of service requirements. The goal of this thesis is to provide tools to evaluate the performance of high speed switching systems. These tools can help with the design of switching systems needed to support current and future communication needs.

The design and analysis of communication systems often require the development and solution of extremely complex models. Three different approaches can be followed in the modeling process: analytical, numerical, and simulation. The main advantage of the analytical approach is that closed form solutions provide explicit relationships between performance measures and input parameters which are very helpful in understanding the behavior of the system. In practice however, closed form solutions can be derived only for extremely simple models with many simplifying assumptions often introduced in the model development. The numerical approach allows the exact or approximate solution of somewhat more detailed models to be computed normally at the expense of significant computational complexity. Simulation can be used for the evaluation of rather detailed models of complex systems whose behavior is investigated in a probabilistic fashion by exploring only some feasible trajectories through the set of possible states.

The primary motivation for this work is the need to evaluate the performance of switching systems using both numerical analysis and simulation. Toward this end we have extended existing queueing analyses that can be applied to switching systems. However, because such analytical models have limited accuracy, and limited generality, the use of simulation is required for a more complete evaluation. Several characteristics make the use of general simulation packages hard or impractical. This led us to the development of a simulator specific for switching simulations.

Another important motivation for the development of a switching simulator was the need for visualization in network performance analysis, which has many advantages over the traditional programming approach. Visualization helps us gain a better understanding of switching systems and provides insight into how the systems actually operate. The simulation tool presented in this thesis was designed to be easy to use and flexible enough to support performance evaluation of a variety of complex switching architectures. Switching system designers and researchers that need to explore design alternatives efficiently, will benefit greatly from such a tool.

A powerful and flexible simulation tool makes it possible to undertake performance studies that are difficult or impossible without such a tool. This thesis compares the performance of several switching systems under the same traffic conditions to obtain a direct side-by-side comparison. We also use the tool to study the transient behavior of some switching systems.

1.2. Thesis Overview

The first chapter after the introduction contains a brief overview of Asynchronous Transfer Mode (ATM) and discusses switching systems in the context of ATM. Some of the issues considered are the functional requirements of ATM switching systems, the topological structure of switching networks used in such systems and the various measures of their performance. Furthermore, we present a taxonomy of ATM switching systems and categorize many different systems that have been proposed in the literature.

The next two chapters consider analytical models of buffered switching networks. Chapter 3 provides methods for analyzing the queueing behavior of copy networks constructed from binary switches. The model is derived for switches employing output buffering and generalized to include switches with input, and shared buffering. The performance results of the copy networks obtained from the analytical models are compared with simulation results and their dependence on fanout

and network size is quantified. The models compare well with simulation for a single switch element. For multistage networks they overestimate the throughput but give a good indication of the dependence of the throughput on the fanout and network sizes, and suggest that copy networks can sustain a high throughput in spite of small buffers.

Chapter 4 provides methods for analyzing the queueing behavior of shared buffer networks. The queueing model is derived under non-uniform traffic conditions. We compare the method with simulation results on the basis of accuracy, where performance is measured in terms of maximum throughput and probability of cell loss.

Analytical models often fall short when applied to the modern high speed communication networks that use highly parallel switching systems to achieve high throughput. To make evaluation and design of such systems easier and to gain a better understanding we present, in Chapter 5, the design and implementation of a general purpose simulation tool for evaluating switching systems. The goal of the design was to create an efficient tool which is general enough to allow performance evaluation of a variety of different switching architectures. The novel aspects of the tool include easy network construction and interaction, visualization and animation of network state and statistics, and fast simulation without compilation.

The use of the simulation tool is demonstrated in Chapter 6 through simulation studies determining the performance of some popular switching systems. First the performance of the systems is obtained for several traffic conditions and compared in terms of throughput, cell loss and delay. Then the performance of the recycling architecture of Turner [65] is studied. Finally we look into the transient behavior of several switching systems, by examining the congestion periods and the time for congestion clearance.

The last chapter summarizes the thesis contributions to the research literature.

2. BROADBAND ATM SWITCHING

This chapter contains an overview of broadband switching within the ATM framework. This overview is used to define terms and present background material. We first briefly discuss ATM and virtual circuit oriented cell switching. Before looking at architectural alternatives for switching system designs we examine the required functionality of switching systems and topological design of multistage switching networks. Furthermore we define the performance measures used to evaluate and compare switching systems.

2.1. Asynchronous Transfer Mode

In the past, separate communication networks have been deployed to support specific services. Today it is desired to design a single all purpose digital communication system, supporting all services in an integrated and unified fashion, including future high bandwidth services. The Asynchronous Transfer Mode (ATM) is widely accepted as the mode of operation for future networks particularly for Broadband Integrated Services Digital Network (BISDN) systems [49].

A key objective of ATM network technology is to ensure consistent performance to users in the presence of stochastically varying traffic. It is necessary to ensure adequate performance for many high speed applications, which require a guarantee on the throughput. To accomplish this it is required that the user requests network resources in advance in order for the network to allocate necessary resources to the anticipated traffic.

ATM networks provide a form of packet switching called virtual circuit oriented cell switching, in which user data is carried in small fixed-length blocks, called cells. Each cell includes a multiplexing label that identifies the user channel that it belongs to. Communication over an ATM network takes place over virtual circuits which are typically established when the user application

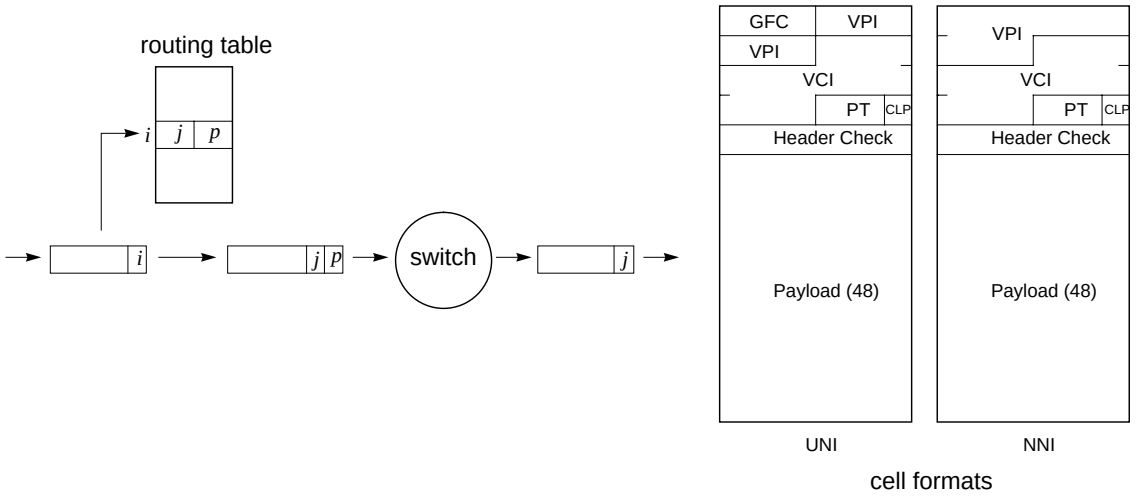


Figure 2.1: Virtual circuit switching and ATM cell format

is initiated. When a virtual circuit is established, a route is selected, and subsequently all cells transmitted on that virtual circuit are forwarded along the selected route.

Virtual circuit oriented cell switching is illustrated in Figure 2.1. The cell's multiplexing label is used to select an entry from a routing table, in order to determine an output port *p* and a new multiplexing label *j*. The cell is written on the output port *j* with the new label. The new label will be used by the next switching unit encountered by the cell. The routing tables have to be set up in advance. The ATM cell formats used at the User Network Interface (UNI) and the Network-Network Interface (NNI) are shown in Figure 2.1. There are two multiplexing options in ATM networks: Virtual Paths, and Virtual Circuits. Cells belonging to different virtual paths are distinguished by their Virtual Path Identifier (VPI) and cells belonging to different virtual circuits are distinguished by their Virtual Circuit Identifier (VCI). User connections are most often implemented using virtual circuits, and virtual paths are essentially bundles of virtual circuits. The VPI and VCI are fields in the five byte header of the ATM cell. Other fields include a Generic Flow Control field (GFC), a Payload Type (PT), a Cell Loss Priority bit (CLP), and Header Check field (HEC). The payload carries user information and has fixed size of 48 bytes [12]. A detailed description of the ATM standard can be found in [49].

2.2. Switching Requirements

The major driving force in the rapid advances in computers and communication has been, fiber optic technology and solid state technology, in particular the development of very large scale integration. The use of fiber optic transmission facilities has made available high bandwidth links which can carry over one terabit of data per second. The processing and switching at the ends of such a facility have become the most important technical challenges facing network designers. ATM line speeds are very high, having rates of 150 Mb/s and 600 Mb/s. Thus the main challenge is to design and build packet switches capable of switching the relatively small cells at these high rates. Fortunately, the progress in the field of VLSI technology has led to new design principles for high performance, high-capacity switching systems to be used in the integrated networks of the future. Most of the proposals for such high-performance switching systems have been based on a principle known as fast packet switching[1]. This principle employs a high degree of parallelism, distributed control, and routing performed at the hardware level.

An ATM switch is a device with N inputs and N outputs which routes the cells arriving on its inputs to their requested outputs. For simplicity we assume that all links have the same transmission capacity and the arrival times of cells at the various input lines are time synchronized. We thus consider time to be slotted, with the slot size equal to the transmission time of a cell on a line, and consider the operation of the switch to be synchronous. There is no coordination among arriving cells as far as their destination requests are concerned. Thus several cells can arrive during the same slot destined to the same output port. Such an event is referred to as an output conflict. Due to output conflicts, buffering of cells within the switch must be provided. Thus the switch provides two functions: routing and buffering. Needless to say, an ideal switch is one that can route all cells from their input lines to their requested output lines with the minimum transit delay possible and with arbitrarily low cell loss, while preserving the order in which they arrived at the switch.

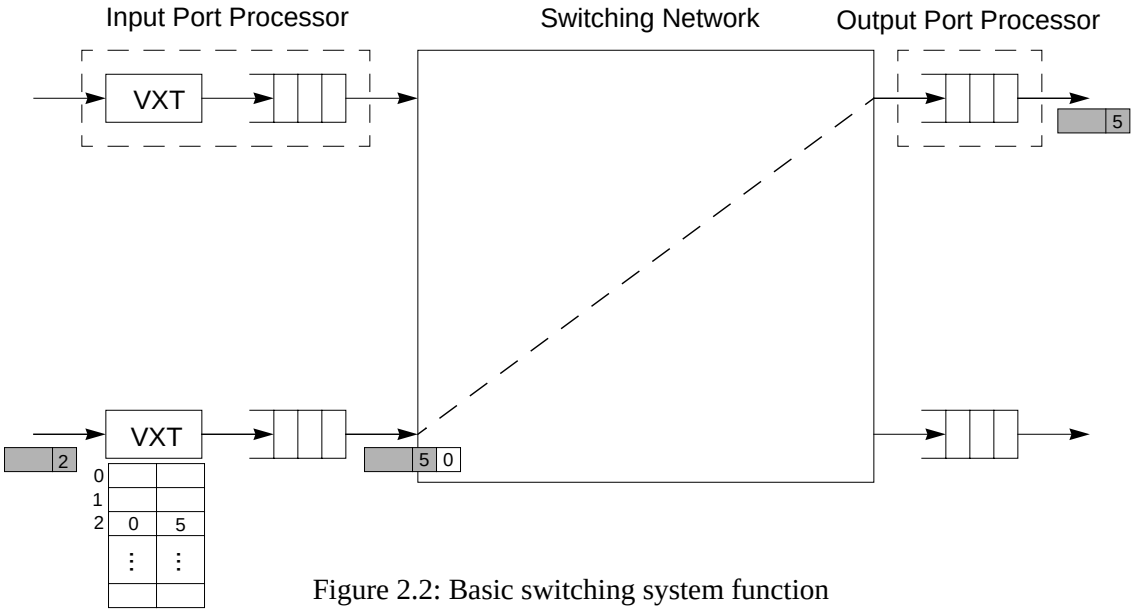


Figure 2.2: Basic switching system function

Besides the basic switching operations that are performed by a switch, in order for multipoint communications to be possible, it is sometimes necessary to send a cell to several output destinations, referred to as multicast. This is accomplished by creating multiple copies of the cell somewhere in the switch, and routing them to the desired output lines.

The basic principle of an ATM switch can be seen in Figure 2.2. Each cell carried on a link has a virtual circuit identifier which identifies the user connection it belongs to. The switch implements a mapping which given an input link and VCI produces an output link address, and outgoing VCI. The Input Port Processor (IPP) performs the virtual circuit translation using Virtual Circuit Translation Tables (VXT) and cell queueing. The Output Port Processor (OPP) also provides some buffering for outgoing cells to absorb short term fluctuations in data rates. The port processors also perform formatting and cell synchronization. A control processor can modify the virtual circuit mapping in response to requests from users. The switching network routes the cells to the requested output ports based on the output addresses from the translation table. Although the basic functionality of an ATM switch is quite simple, it is rather challenging to design switching networks that meet the speed requirements discussed earlier, with low cost and scalability.

2.3. Switching Network Topology

Before we discuss the various switching architectures we first review the topology and construction of switching networks. The term network is used in the remainder of the thesis to refer to a switching network, which connects together a set of inputs to a set of outputs. The simplest example of a network that connects N inputs with M outputs is a crossbar switch, represented in Figure 2.3 in two different ways. The crossbar corresponds to an $N \times M$ array. Semiconductor switches are located at each of the crosspoints where inputs and output wires cross. We connect an input to an output by closing a crosspoint at the intersection of the appropriate row and column. The complexity of a crossbar has two cost components, one which grows in proportion to the number of inputs and outputs and the other that grows as their product. The product term is often called the crosspoint count because it is directly related to the number of simple 2×2 crosspoints required to implement it. A crossbar requires N^2 crosspoints for N paris of terminals. We can reduce this complexity by using several smaller crossbars to construct a multistage network.

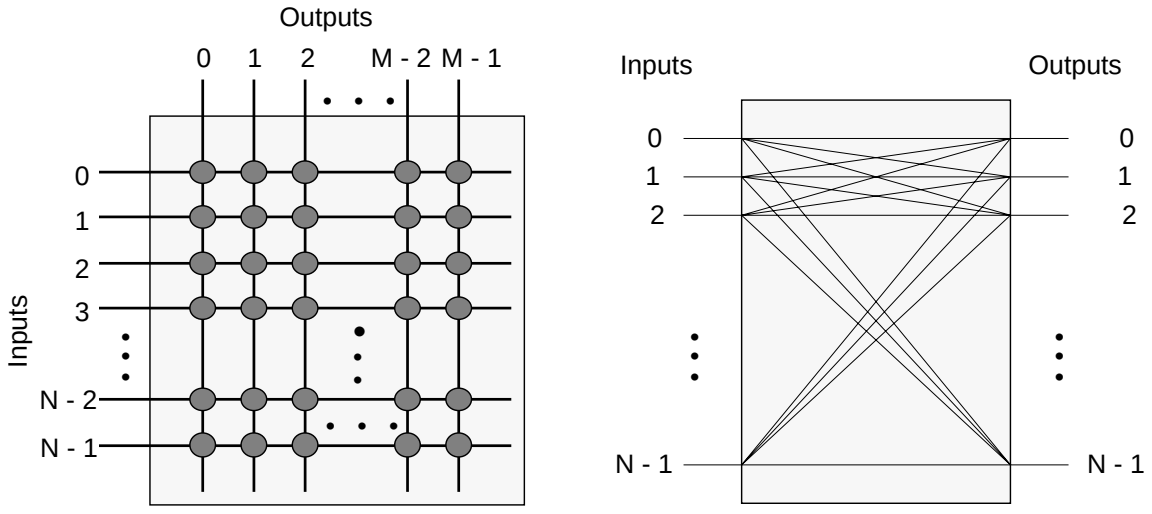


Figure 2.3: Representations of a crossbar switch

A network's topology can be described by a graph model. The model used here follows the terminology of Turner and Melen [42]. A network is denoted by a quadruple (*S*, *L*, *I*, *O*), where *S* is a set of vertices, called switches, *L* is a set of arcs called links, *I* is a set of input terminals and *O* is a set of output terminals. Each link is an ordered pair (*x*, *y*) where $x \in I \cup S$ and $y \in O \cup S$. Input and output terminals can only appear in exactly one link. Links which include an input terminal are called inputs and those including an output terminal are called outputs. The remaining links are internal. A network with *n* inputs and *m* outputs is referred to as an (*n*, *m*)-network. An (*n*, *n*)-network is also called an *n*-network.

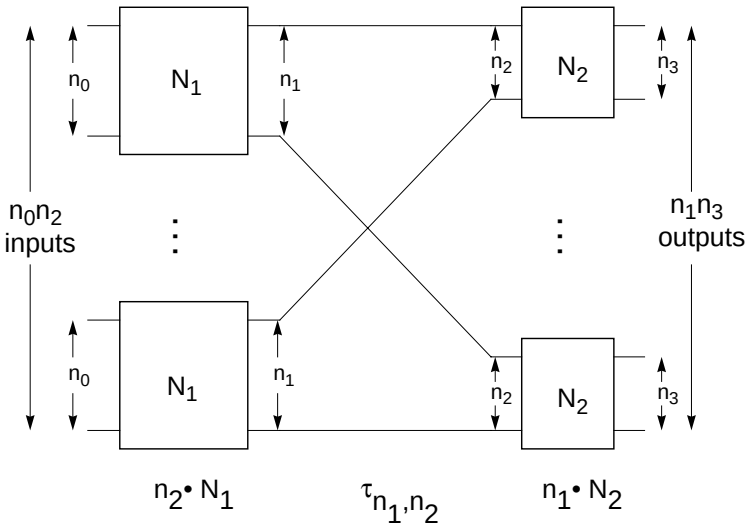


Figure 2.4: Series connection of networks

The depth of a switch or output terminal *x* is defined as the number of arcs in a shortest path from an input terminal to *x* and the depth of input terminal is defined to be zero. A graded network is one in which all arcs in the graph model go from vertex *x* to vertex *y*, where the depth of *y* is one greater then the depth of *x*. In this case a switch is said to be in stage *i* if it has depth *i* and a link is said to be in rank *i* if its depth is *i*. When the outputs are in stage *k*, then the graded network is

called a *k*-stage network. Inputs and outputs of a network are numbered top-down starting from 0. Switches are labeled by an ordered pair (*i, j*) where i is the stage number of the switch and *j* is a row number identifying the switch in that stage. Each switch has a set of numbered ports with which its incident links are identified.

The topology of many networks can be described in a systematic way by using several operators and components. There are two basic components: an *m* input, *n* output crossbar switch, denoted $X_{m,n}$, and a subnet defined by a permutations on $\{0, \ldots, n-1\}$. Basically any permutation can be used; however, the one we use extensively is the generalized perfect shuffle, which can be described by *n* items divided into groups of *a* items, and then interleaved. More precisely, if *a* and *b* evenly divide *n*, then $\tau_{a,b}$ is defined to be the permutation that satisfies

$$\tau_{a,b}(ja + i) = ib + j \qquad 0 \le i \le a-1, 0 \le j \le b-1$$

The variables are all integers.

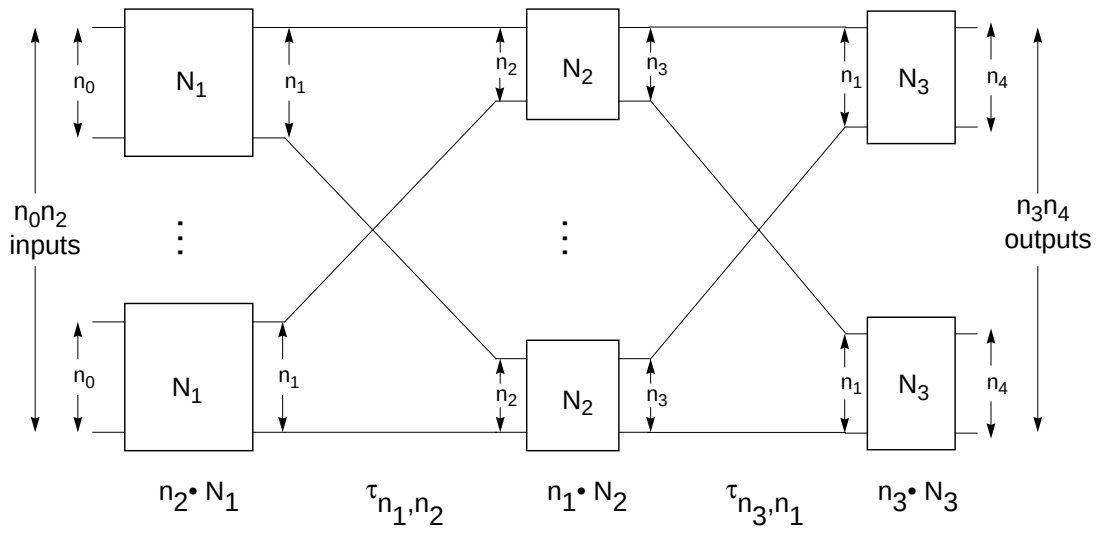


Figure 2.5: Parallel connection of networks

Networks can be constructed using several operations. One such operation is the concatenation of two networks N_1 and N_2 denoted $N_1;N_2$ and obtained by connecting output i of N_1 with input i of N_2. If i is a positive integer and N an (n, m)-network, then $i \cdot N$ denotes the network obtained by taking i copies of N, without interconnecting them. The reverse of a network N is denoted N' and is obtained by exchanging inputs and outputs and reversing the directions of all links. If N_1 is a network with n_1 outputs and N_2 is a network with n_2 inputs, then the series connection of networks N_1 and N_2 is denoted as $N_1 \times N_2$ and is defined as:

$$N_1 \times N_2 = (n_2 \cdot N_1);\tau_{n_1, n_2};(n_1 \cdot N_2)$$

The series connections is shown in Figure 2.4, where it can be seen how the permutation τ_{n_1, n_2} connects every output of a particular copy of N_1 to a different copy of N_2 and that there are as many copies of N_1 as there are inputs to N_2. Thus, if N_1 is an (n_0, n_1)-network and N_2 is an (n_2, n_3)-network then the resulting network will be an (n_0n_2, n_1n_3)-network.

Another important construction operator is the parallel connection which combines three networks. If N_1 is an (n_0, n_1)-network, N_2 is an (n_2, n_3)-network and N_2 is an (n_1, n_4)-network then

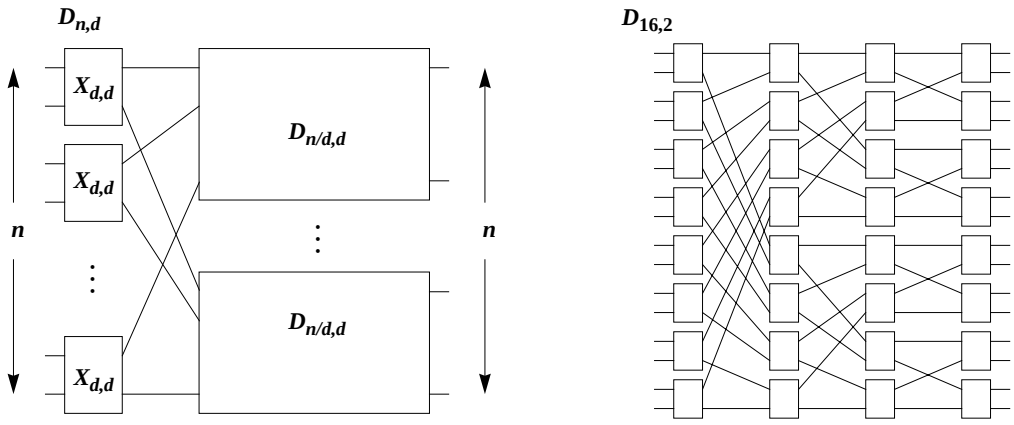


Figure 2.6: Definition of delta network, $D_{n,d}$.

the parallel connection is defined as:

$$N_1 \otimes N_2 \otimes N_3 = (n_2 \cdot N_1) ;\tau_{n_1, n_2}; (n_1 \cdot N_2) ;\tau_{n_3, n_1}; (n_3 \cdot N_3)$$

This network can be thought of as two independent series connections, first a series connection of N_1 with N_2 and then a series connection of N_2 with N_3. The construction shown in Figure 2.5 results in a ($n_0 n_2$, $n_3 n_4$)-network.

The construction operators discussed above can be used to describe many networks, two of which are discussed in some detail here. The delta network [22] with n inputs and constructed out of $d \times d$ switches is denoted by $D_{n, d}$ and is defined recursively using series connections as:

$$D_{n, d} = X_{d, d} \times D_{n/d, d} \qquad D_{d, d} = X_{d, d}$$

where $X_{d, d}$ denotes a $d \times d$ switch. Figure 2.6 shows the recursive construction of the network and an example of a $D_{16, 2}$ network. The delta network has $\log_d n$ stages and is isomorphic to topologies such as the banyan [29] and the omega [36] networks.

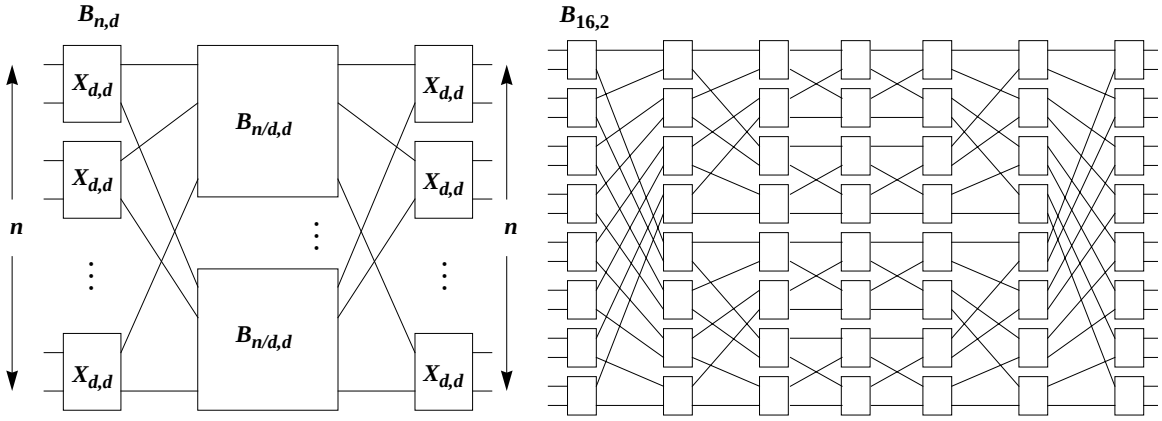


Figure 2.7: Definition of Benes network, $B_{n,d}$.

The Benes network [6] with n inputs and constructed from $d \times d$ switches is denoted $B_{n, d}$, and is defined recursively using parallel construction:

$$B_{n, d} = X_{d, d} \otimes B_{n/d, d} \otimes X_{d, d} \qquad B_{d, d} = X_{d, d}$$

Figure 2.7 shows the recursive construction of this network and as an example the $B_{16,\,2}$ network. Other well known networks constructed using the network construction operators described are summarized in Table 2.1. As can be seen from the table, these few simple construction operators are sufficient to build a wide variety of networks with different topology and complexity. The complexity is shown as a number of crosspoints per input. Although the crosspoint measure is simple and gives a good idea about relative network complexity it should not be interpreted as indicating precise cost, since the cost associated with the measures is technology dependent. For networks constructed from large scale integrated circuits a better measure is the number of integrated circuit packages [70]. In cases were networks, or portions of networks, can be placed entirely on a single integrated circuit package, the area taken up by the circuit becomes the most important complexity measure. Furthermore, the number of pins can limit the number of signals that can enter or leave a physical component. These cost measures depend on technology and implementation.

Table 2.1: Construction and complexity of common switching networks

Network	**Construction**	**Crosspoints/input**
3-Clos	$C^{3}_{n,\,d,\,m} = X_{d,\,m} \otimes X_{\frac{n}{d},\,\frac{n}{d}} \otimes X_{d,\,m}$	$2m + \frac{nm}{d^2}$
Optimal Clos	$C^{*}_{n,\,d,\,2d} = X_{d,\,2d} \otimes C^{*}_{\frac{n}{2},\,d,\,2d} \otimes X_{d,\,2d} \qquad C^{*}_{d,\,d,\,2d} = X_{d,\,d}$	$d\left(\frac{5}{2}n^{\log_d 2} - 4\right)$
Banyan	$Y_{n,\,d} = \tau_{\frac{n}{d},\,d};(X_{d,\,d}) \times Y_{\frac{n}{d},\,d} \qquad Y_{d,\,d} = X_{d,\,d}$	$d\log_d n$
Benes	$B_{n,\,d} = X_{d,\,d} \otimes B_{n/d,\,d} \otimes X_{d,\,d} \qquad B_{d,\,d} = X_{d,\,d}$	$d\,(2\log_d n - 1)$
Bitonic sorter	$S_n = \left(2 \cdot S_{\frac{n}{2}}\right);Y_n \qquad S_2 = X_{2,\,2}$	$\log_2 n + \log_2^2 n$
Cantor	$K_{n,\,d,\,m} = X_{1,\,m} \otimes B_{n,\,d} \otimes X_{m,\,1}$	$2m\left(d\log_d n - \frac{d}{2} + 1\right)$

Table 2.1: Construction and complexity of common switching networks

Network	Construction	Crosspoints/input
Delta	$D_{n,d} = X_{d,d} \times D_{n/d,d} \qquad D_{d,d} = X_{d,d}$	$d\log_d n$
Ext. delta	$D^*_{n,d} = D_{d^h,d} \otimes D_{d^{k-h},d} \otimes D'_{d^h,d}$	$d\,(h + \log_d n)$

2.4. Performance Considerations

The performance of switching systems is mainly characterized by the following measures: the throughput, the connection blocking probability, the probability that cells get lost, the switching delay and the jitter on the delay. We discuss these measures in what follows.

The switch throughput is the traffic carried by the switch expressed as a utilization factor of its output links, and defined as the probability that a cell leaves a switch output in a particular slot. The maximum throughput, also referred to as switch capacity, indicates the load carried by the switch for the maximum offered load. A large overall switch throughput can be achieved by a proper topology and architecture.

Because ATM is connection oriented, resources needed to support the connection must be available at connection setup. In the case when there are not enough resources available we say that the connection is blocked. A useful measure, termed connection blocking, is defined as the probability that not enough resources can be found between the requested inlet and outlet of the switch to guarantee the quality of all existing connections plus the new connection. The blocking probability depends on the topology, the dimensions of the switch elements, the speed advantage of the internal links compared to the external links, the number of internal connections and their load [67]. Switching systems can be designed to be internally non-blocking at the cost of additional complexity.

In ATM switches, it is possible that too many cells are destined for the same link. The consequence is that the number of cells that simultaneously compete for a particular queue in the switch

exceeds the number of cells the queue can store, resulting in cells being lost. The probability of losing a cell must be kept within acceptable limits. Some switches can only lose cells at the switch's inlets or outlets and not internally in the switching fabric. It is also possible that cells belonging to the same connection will get out of order. Since ATM systems cannot deliver cells out of order, the misordered cells have to be restored to the proper order or dropped. If cells are dropped the probability of loss must be kept within the cell loss limit. The cell delay is defined as the number of slot periods from the time a cell is received at the input until it is transmitted at the output. The delay consists of a fixed switching delay which depends on implementation and the queueing delay that heavily depends on the load of the links and the size and the occupancy of the queues. The term jitter refers to the variability of the cell delay about the mean.

An important factor affecting the performance of a switch is the traffic pattern according to which cells arrive at its inputs. The traffic is described by the arrival of cells at the inputs of the switch, and the destination requests of the arriving cells. The simplest traffic pattern of interest is referred to as independent uniform traffic pattern. The. arrival of cells at an input line is a Bernoulli process with parameter p, independent from all other input lines, and whereby the requested output port for a cell is uniformly and independently chosen among all output ports. Other traffic patterns may exhibit dependencies in the cell arrival processes as well as in the distribution of output ports requested. For example, cells may arrive at an input line in the form of bursts of random lengths, with all cells in a burst destined to the same output port. The traffic pattern in this case is defined in terms of the distributions of burst lengths, of the gap between consecutive bursts, and of the requested output port for each burst. Such a traffic pattern may be referred to as a bursty traffic pattern. Yet another example of time-dependent traffic pattern may be found in applications that produce cells at regular intervals such as constant bit rate traffic.

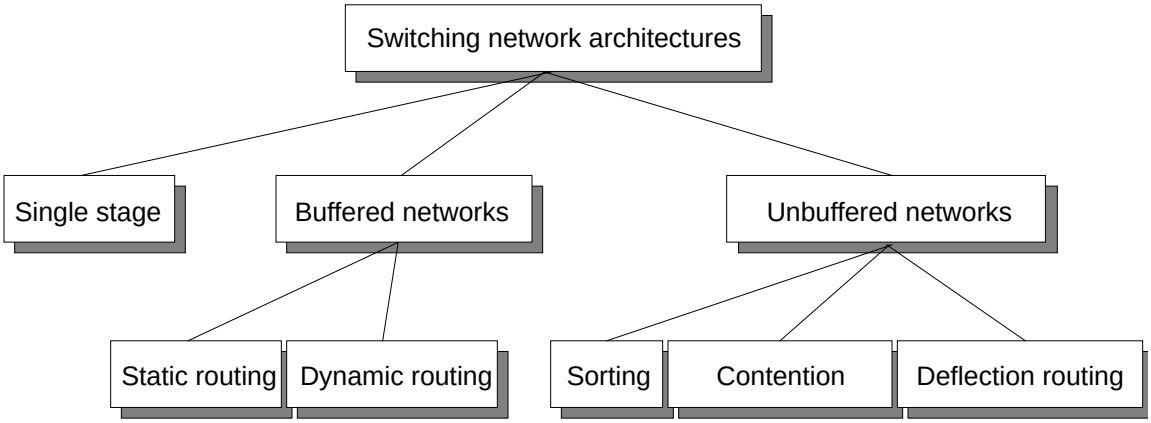


Figure 2.8: Taxonomy of ATM switching systems

2.5. Switching Network Architectures

The performance characteristics of switching systems depend heavily on the architecture used in the system. In this section we discuss some of the wide variety of ATM switching architectures that have been proposed [1][55]. A unique taxonomy is difficult to find, since different properties used in various order can be used to classify ATM switches. The classification presented here shown in Figure 2.8 is based on the structure of the switching system, the routing methods, and the resolution of output contention. The various types of ATM switching architectures are discussed in the following subsections.

2.5.1. Single stage

Switching systems using networks consisting of a single stage are relatively simple. The connectivity between the input or output port processors is provided by a crossbar or a shared high speed medium like a bus or a ring. In such a configuration the output port contention is resolved with some arbitration scheme. This leads to the simple structures shown in Figure 2.9, where the size of the systems is limited by the bandwidth of the shared medium. Furthermore, as the number of ports increases the complexity of such systems grows very quickly.

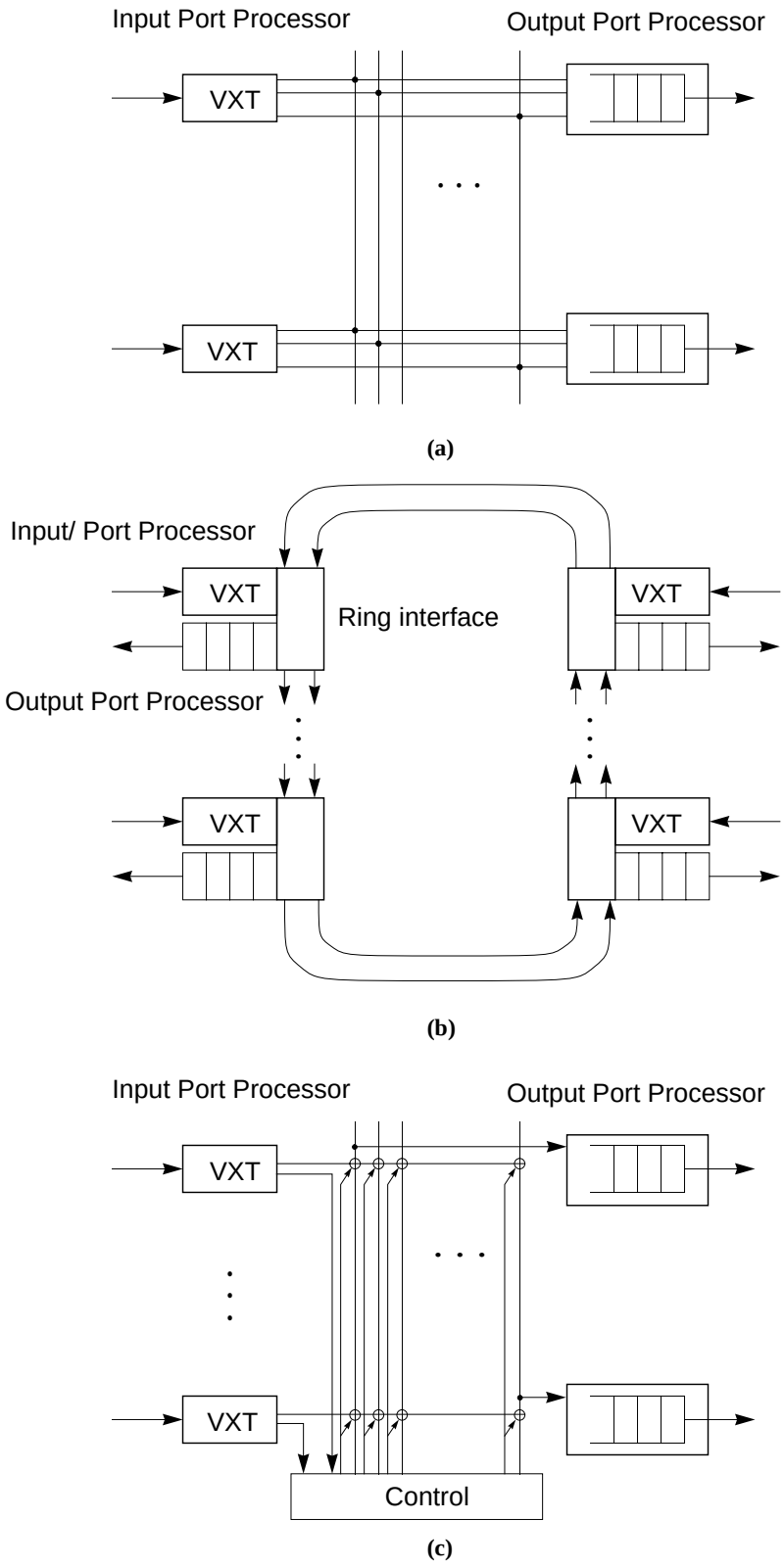


Figure 2.9: Single stage switching systems. Bus (a), ring (b) and crossbar (c)

Bus based switching system. In such a system shown in Figure 2.9 (a) a common bus provides the connectivity among port processors. The bus can typically be used by only one port processor at a time. For an n port system, with links of R bits per second, and a clock rate of r ticks per second, a bus width of greater than nR/r bits is needed. In addition to limitations on system size by bus bandwidth, the capacitive load limits the number of taps on the bus. Since every processor must interface to the bus at full rate, extension to larger input/output sizes requires added parallelism leading to quadratic growth in complexity.

Ring based switching system. In this configuration shown in Figure 2.9 (b) rings eliminate the capacitive loading of buses, giving them an advantage at very high speeds, but increasing the minimum latency. Such systems have the same bandwidth requirements and complexity characteristics as the bus based systems.

Crossbar based switching system. Crossbars also avoid the capacitive loading of busses and reduce the width of the data path needed at the port processors. The crossbar retains quadratic complexity as systems grow, but because the quadratic cost component is concentrated within the crossbar, it reduces the system cost. An additional advantage is that the crossbar can be used by multiple port processors at the same time. For an n port system with links of R bits per second, and a clock rate of r ticks per second, the required port processor interface width is R/r bits instead of the nR/r bits required for the bus and the ring configurations. If the crossbar operates at the same speed as the links, contention for outputs can cause congestion at input port processors which is referred to as head of line blocking. Maximum throughput of such systems is about $0.58n$ [72] cells per cycle [19] leading to a typical data path of $2R/r$ bits. Alternate approaches are serial bypass queueing, which allows an unsuccessful input port processors a chance to contend again. Parallel bypass queueing provides each port with multiple crossbar inputs allowing it to submit more than one cell on each input in a given cycle. Finally it is possible to maintain the data path

width of *R*/*r* bits on input size but provide *n* crossbar outputs per output port, feeding a knockout concentrator. An example of such a switch design is the Knockout Switch [72].

2.5.2. Unbuffered networks

As indicated by the name, unbuffered networks have no buffering within the switch fabric. Multistage networks made from simple switching elements are usually used resulting in a complexity that is much lower than the complexity of a single crossbar. The networks are classified based on their contention resolution in three different types: Contention based, networks using deflection routing, and based on sorting.

Contention based networks. Many of the well known interconnection networks such as omega, flip, cube, shuffle-exchange and baseline are all isomorphic to the delta networks [46]. These networks have been considered for ATM switching systems. They can be constructed in a modular way from smaller *d* port switching elements and have a self routing property for cell movement from any input to any output which uses *d*-ary destination addresses. Each switch element extracts routing information from the cell header, performs local arbitration, and allows winners to proceed to the next stage. While these networks are capable of switching cells simultaneously in parallel, they are cell blocking networks in the sense that cells can collide with each other and get lost as demonstrated in Figure 2.10. To avoid cell loss an acknowledgment has to be sent through the upstream acknowledgment path so that the losing cells can be transmitted again in the next cycle. The contention results in a reduction of the maximum throughput of the switching system, especially when the traffic is non-uniform. There are several ways to reduce the blocking and to increase the throughput such as: increasing the internal link speeds relative to the external speeds, using multiple networks in parallel to provide multiple paths from any input to any output, replacing the internal link with multiple links for each switch element connection,

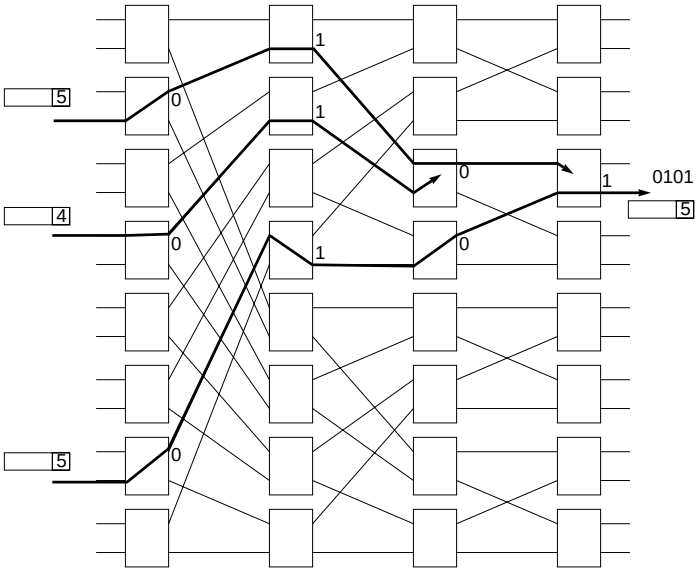


Figure 2.10: Contention in an unbuffered delta network.

resulting in what is referred to as a dilated network, and finally using distribution stages in front of the network to distribute the load evenly.

Networks using deflection routing. Deflection routing is a method used to handle situations in which more than one cell is destined to the same output of a switch element. In deflection routing, one cell is sent to the desired output while the other is deflected to a different output. The cells that are deflected to wrong outputs are given chances later to reach their destination through some alternate path. To improve the probability that deflected cells will reach their destinations additional stages are used in the network. Those that do not reach their destination by the final network stage are lost or can be buffered and given another chance by recirculation. Several switching systems use this method, among them, the tandem banyan and Shuffleout networks.

The Tandem Banyan network [56] consists of *k* banyan networks arranged one after the other as shown in Figure 2.11 (a). The outputs of each banyan are connected to the next banyan and to an output buffer associated with an output port. Each cell is routed through the banyan networks

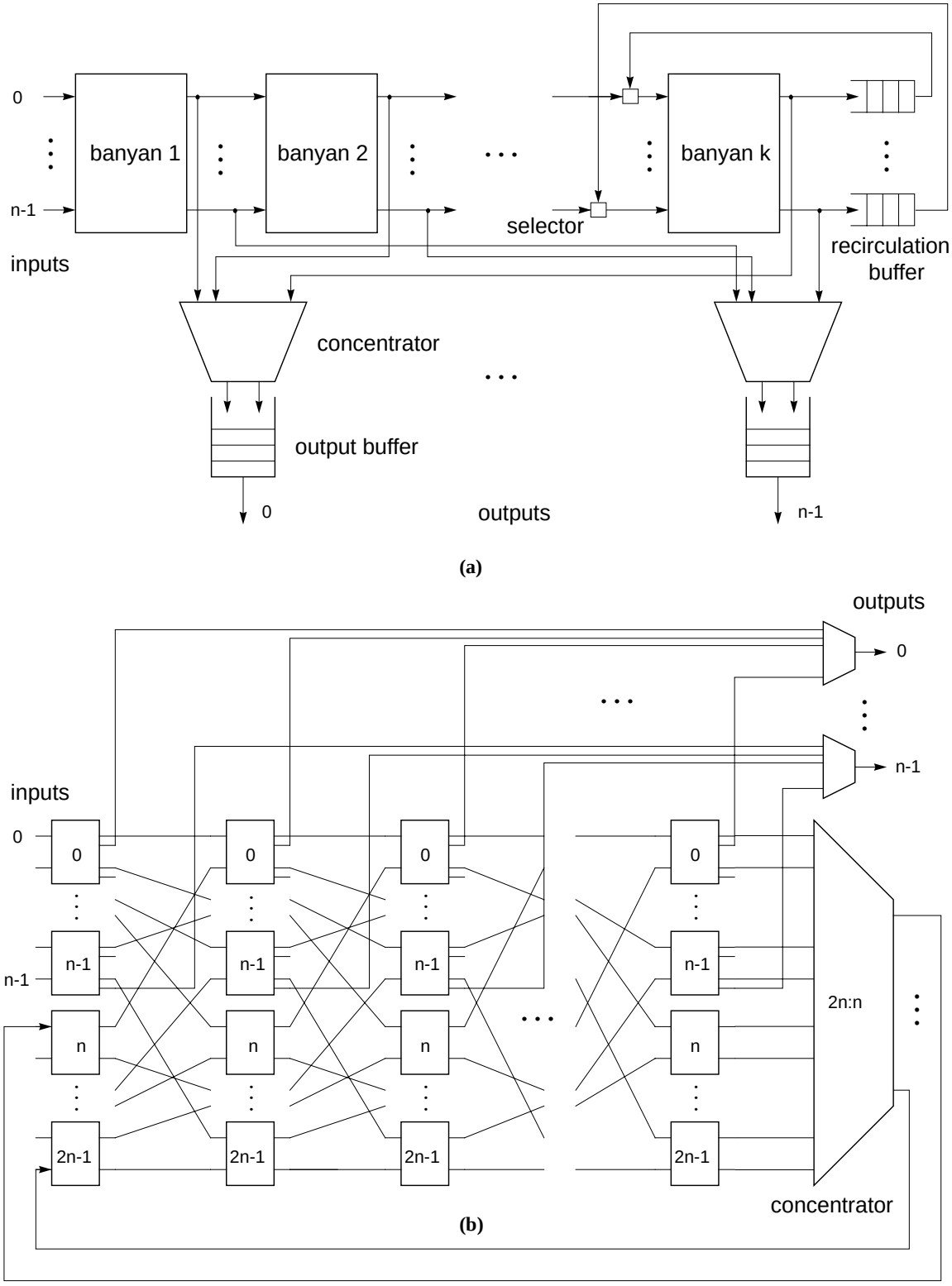


Figure 2.11: Deflection routing networks, Tandem Banyan (a) and Shuffleout (b).

towards the requested destination output. When conflicts occur between cells that require the same output of a switch element, one cell goes to the desired output but the other is marked as misrouted and deflected off the right path. When a cell reaches an output of the banyan network, it is either sent to the associated output buffer, or if it has been deflected it is unmarked and sent to the next banyan network where it will contend again for the output port. The last banyan network can either discard marked cells, or be connected to a set of recirculation buffers which feed back to the inputs of the last network through a selector, or send flow control signals back through the network. Delay elements ensure that cells arrive at the output buffer at the same time in correct order, however recirculation can cause cells to arrive out of order.

The Shuffle network [20] has a shuffle pattern interconnection resulting in an omega type network shown in Figure 2.11(b). It consists of 2x4 switch elements, each with two of the outputs going to output ports and the other two connecting to the next stage. If a cell has reached a switch connected to the destination port then the cell is sent to the output port, otherwise it is sent to the switch element in the next stage that is closest to the desired output port depending on a distance function. If two cells conflict, the losing cell is deflected to the wrong output and tries to reach its destination in the following stages. Cells that leave the last stage without success are recirculated into the network at the first stage through dedicated recirculation ports. The number of recirculation ports is chosen so that cell loss at the concentrator is tolerable.

Sorting based networks. The basic idea behind a sorting based network is to avoid cell blocking completely by first sorting cells based on their destination addresses and then routing them through a banyan network. Such networks are often referred to as Batcher-banyan networks [4]. These networks exploit the fact that a banyan network is nonblocking if the cells arrive at consecutive inputs and destined for outputs in increasing order. Figure 2.12 shows a Batcher sorting network which is constructed from 2 x 2 elements. Each element is a comparator that compares the

addresses of the incoming cells and places the cell with smaller address value at the output indicated by the arrow head. The cell with the larger address value is routed to the output indicated by the arrow tail. An n input sorting network S_n, is defined recursively from subnetworks called bitonic merge networks, denoted M_n. The merge network has the property that given a bitonic sequence at its inputs, it will produce a sorted sequence at its outputs. A sequence of numbers is bitonic if the sequence can be divided into a non-decreasing sequence followed by a non-increasing sequence. To construct S_n, two copies of $S_{n/2}$ are connected to a merge network M_n as shown in Figure 2.12. The bitonic merge network is simply a binary banyan network.

As long as no two cells are destined to the same output the sorting based networks are non-blocking. Thus, resolution of the contention between cells destined for the same outputs is an issue that needs to be dealt with. Several different methods employed in these networks are shown in Figure 2.13 (a). The first method uses an arbitration ring that has one bit for every output [9]. This

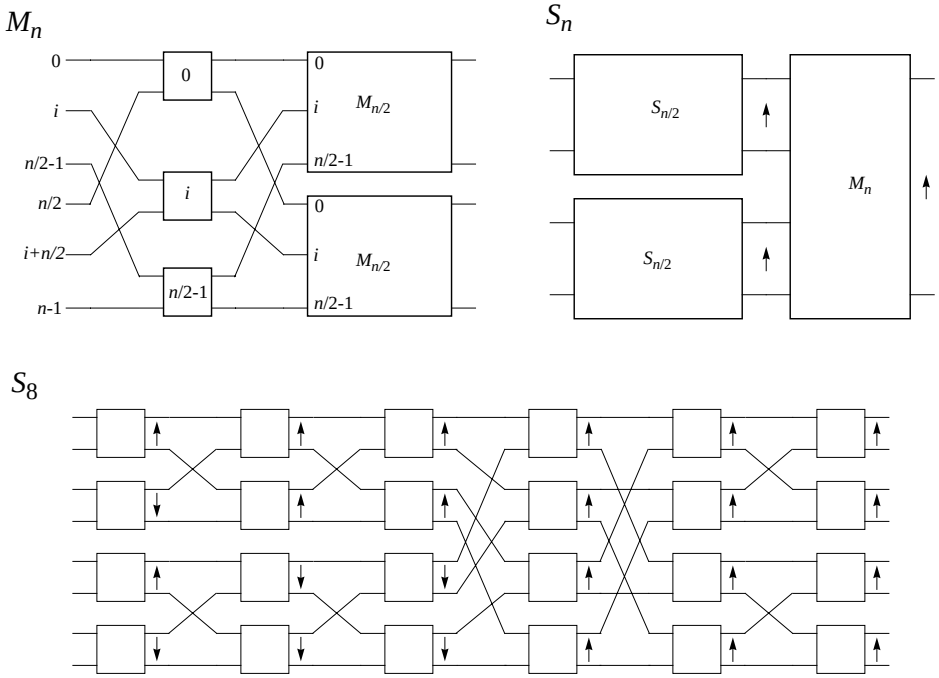


Figure 2.12: Sorting and a merge network

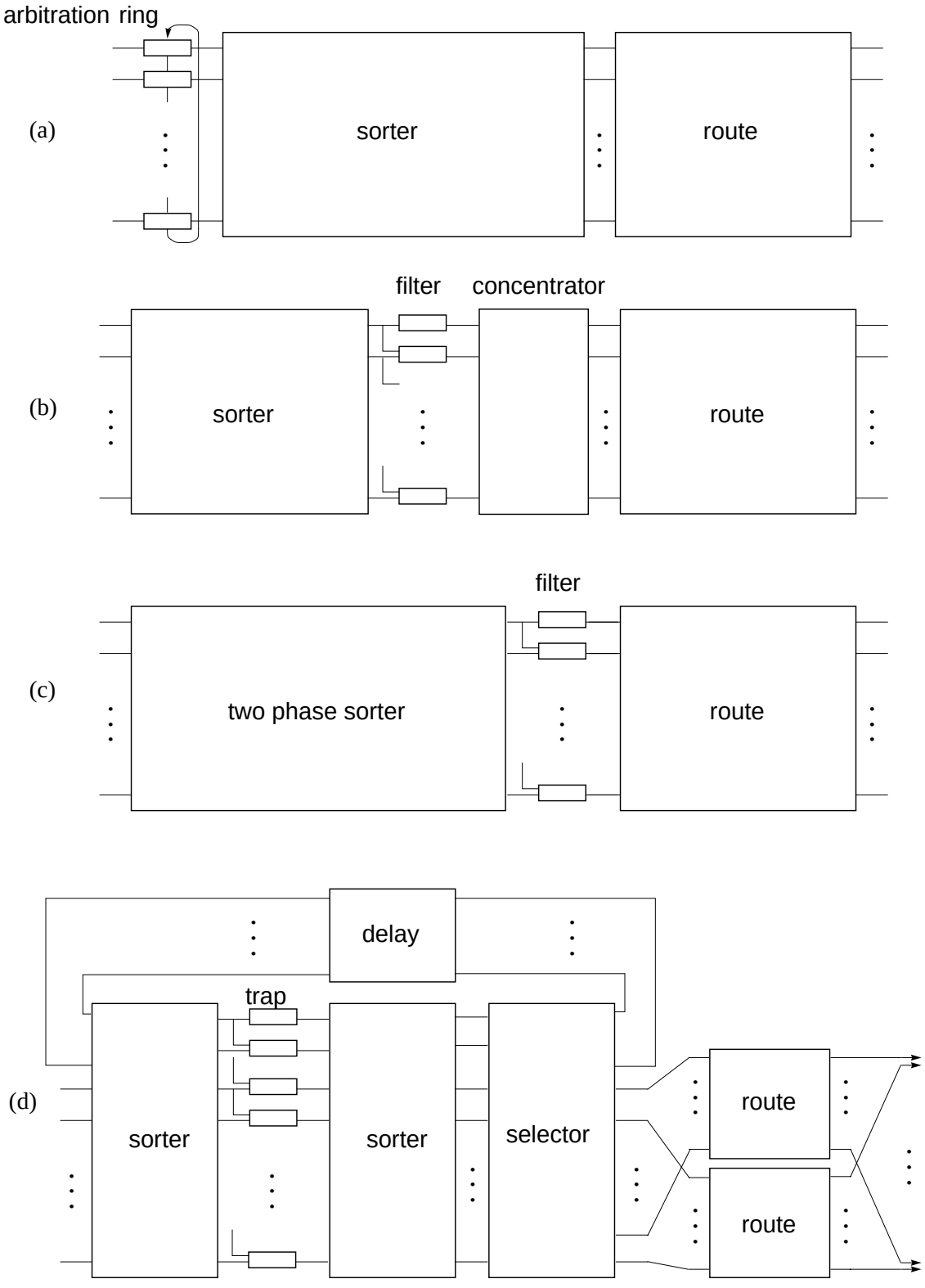


Figure 2.13: Arbitration for sorting based networks

ensures that each output receives at most one cell. The arbitration ring must circulate within one operation cycle, putting an upper bound on switch size. A more complex method, which allows one to build larger switches, uses filters after the sorting network to remove multiple cells destined for particular outputs, as shown in Figure 2.13 (b). The filters accomplish this by making local comparisons to identify winners. A concentrator takes the winning cells and puts them on consecutive outputs so that the banyan routing network can route them without conflicts. Another method for output contention resolution shown in Figure 2.13 (c), uses a two phase operation. The two phases are a contention phase and data phase. During the contention phase the inputs send output requests to the network and the sorting network sorts the requests and filters detect duplicate outputs and acknowledge the winners. During the data phase the winners transmit their cells which pass through a sorting network and a banyan routing network without conflict. The drawback of this method is the extra overhead required for the contention phase. Finally systems use a recirculation of duplicate cells as a method to resolve contention between cells destined for the same output. An example of such network is the Sunshine switch [27]. In this network a sorter orders cells by priority and destination. Then, a set of trap filters mark the extra cells that are duplicates for recirculation. A second sorter separates winners from losers and sorts the losers by priority. A selector routes the highest priority losers to the recirculation ports and passes the winners to the routing network which guides cells to the outputs. Multiple banyan networks can be used to allow more than one cell destined for the same output to go through the network per cycle. This reduces the number of recirculation ports needed.

2.5.3. Buffered networks

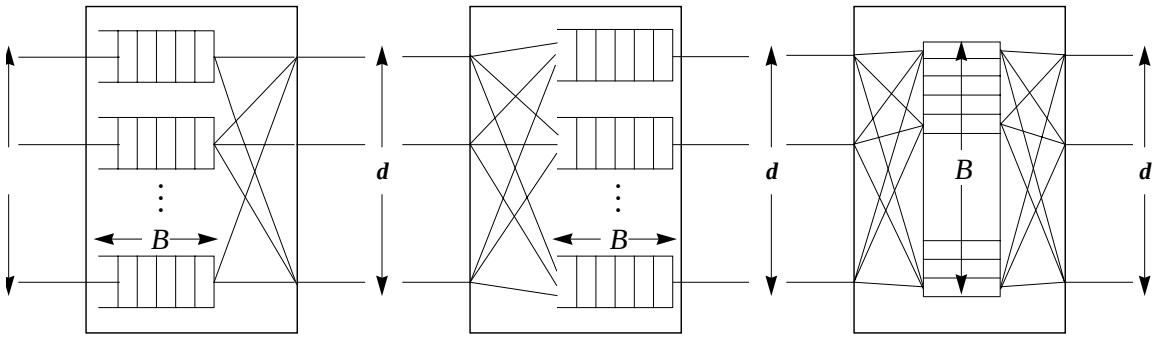
To improve the throughput of multistage switching networks one can place buffers in each of the switching elements. This idea was first introduced and analyzed by Dias [22] and considered

for a general-purpose packet switching system by Turner [58]. When two or more cells conflict in a switching element, one of the conflicting cells is forwarded to the output port but the other cells remain in a local buffer. Three different buffering options are shown in Figure 2.14 (a). FIFO input queueing can limit performance due to head of line blocking. To improve the performance to the same levels as output queueing, bypass input queuing can be used, but both require high speed access rates to the buffers. Shared buffering allows a smaller set of buffers to be shared by all inputs and outputs, but it requires a crossbar or bus on both sides of the buffer.

To ensure that the buffers in the switch elements will not overflow, multistage networks use flow control between switch elements in different stages. The flow control prevents a switch element from receiving a cell when its buffers are full. There are two kinds of flow control: Grant and acknowledgment. Grant flow control gives permission to an upstream switch element to send a cell. Acknowledgment flow control acknowledges that a cell has been received and safely stored in the buffer. Flow control can be either global or local. Global flow control depends on both the state of a switch element and the flow control signals from downstream neighbors. For local flow control, the signals only depend on the local state of the switch element.

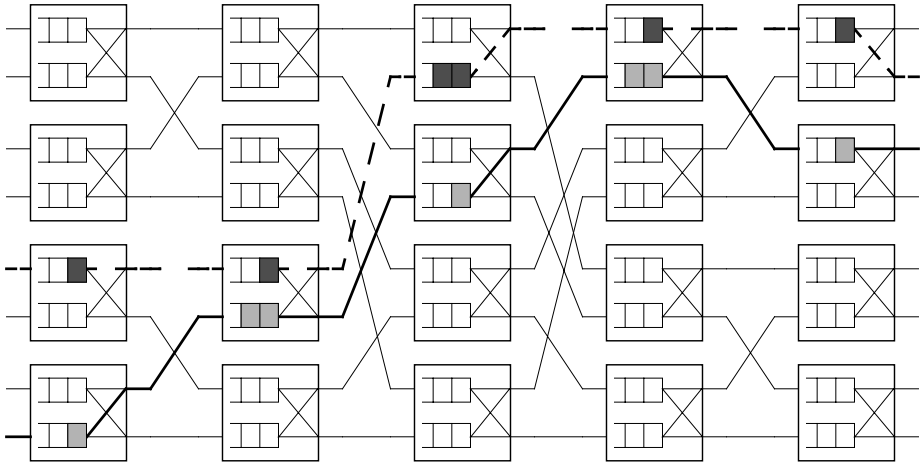
Buffered networks can use different buffering mechanisms and flow control, but the most important aspect that distinguishes different networks is the way cells are routed in the network. The two approaches used are static routing and dynamic routing.

Static routing. Networks that route all cells belonging to the same connection along the same path through the multistage network, are said to be static routing networks. During a connection setup an available route through the network is determined for the connection and all cells are then routed by that path. If there is no route with enough resources available in the network from the input to the requested destination, the connection is blocked. The performance of such networks is therefore determined by the probability of blocking in addition to the queueing in the switching

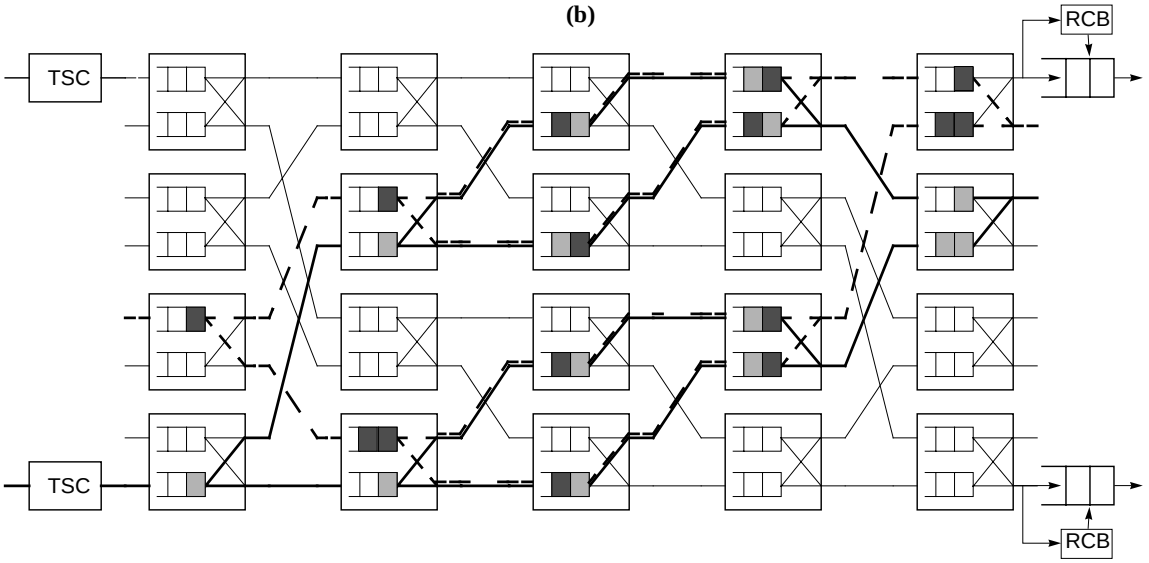


input buffer **output buffer** **shared buffer**

(a)



(b)



(c)

Figure 2.14: Buffering in switch elements (a), static routing network (b), dynamic routing network (c).

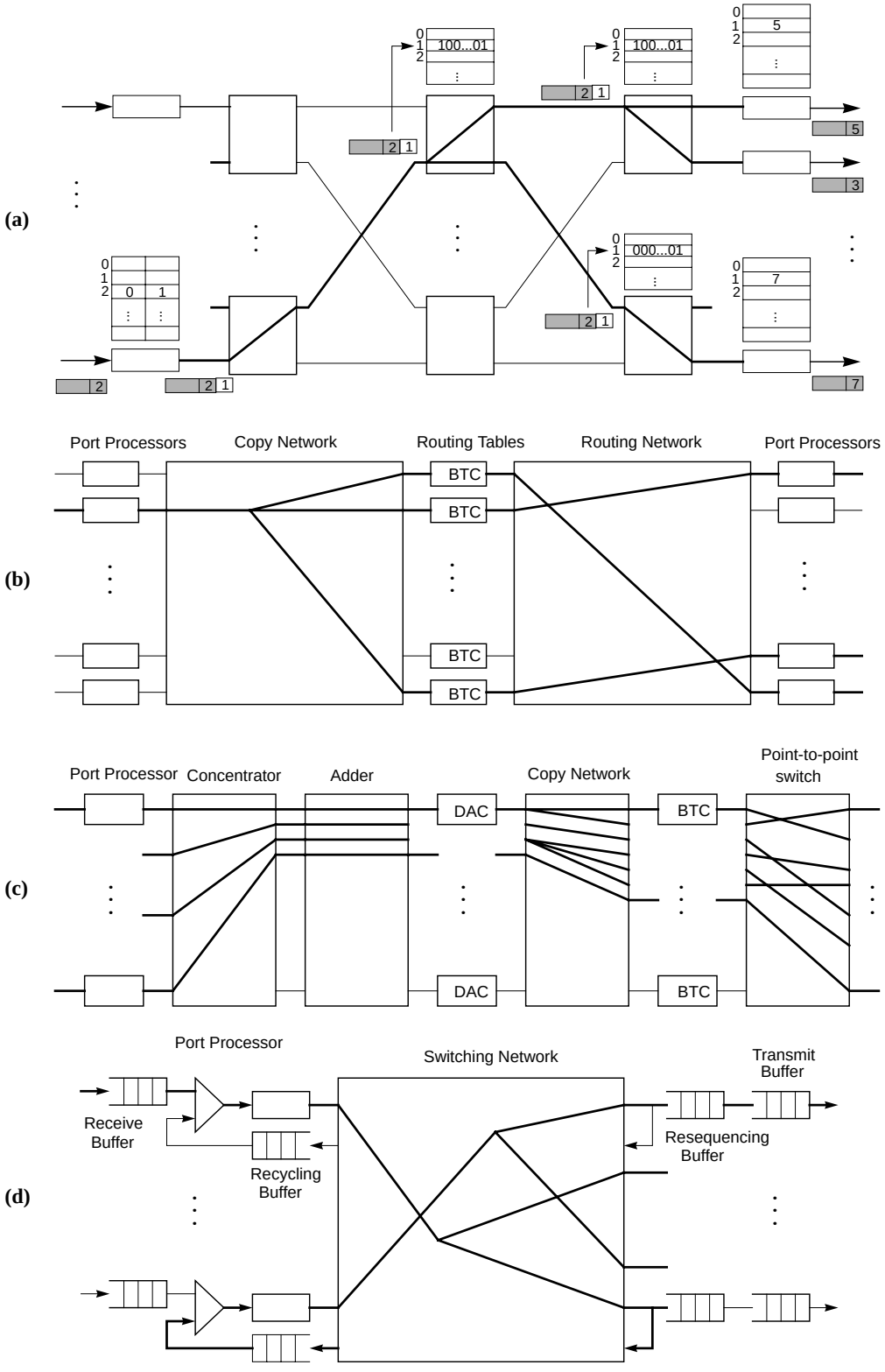
system. With the right combination of topology and speed advantage, static routing networks are non-blocking, that is, there is always a path available in the network to satisfy a legitimate connection request [42]. The design of the switch elements and choice of the internal data path speed determine the queueing characteristics. An example of a static routing switching system is the Atom switch [53] which uses output buffered switch elements.

Dynamic routing. The blocking that occurs with static routing networks can be avoided with less complexity if cells are routed individually. The cells can follow any of the multiple routes in the network that leads to the requested destination. This is referred to as dynamic routing and it allows a more uniform distribution of the load throughout the network, reducing the speed advantage needed for the network to be non-blocking compared with the static routing networks. One disadvantage of dynamic routing is that cells travelling different paths can encounter different delays in the intermediate queues and may arrive at the destination port out of order making it necessary to put the cells back in sequence at the outputs. Resequencing can be done with a time stamp circuit (TSC) at the input and a resequencing buffer at the outputs. The buffer control (RCB) keeps track of the age of the cells in the buffer via the time stamp and always selects the oldest cell for transmission as long as the cell is older than a certain threshold. If a cell arrives at the buffer that is older than the age threshold it is discarded. The age threshold is determined by the network's internal storage capacity and delay distribution. By selecting the size of the buffer and the age threshold it is possible to reduce the probability that cells are misordered down to the same levels as the probability of cell loss in the network. The speedup advantage of the network depends on the network topology and can be approximated by using a fluid flow analysis discussed in [59]. Examples of switching systems that fall in this category are the Broadcast Packet Switch [61], and the Recycling switch [65].

2.6. Multicast Networks

An application such as video conferencing may require multipoint connections joining an arbitrary number of endpoints. A multipoint connection requires that a cell originating at a terminal be delivered to more than one destination. This could be accomplished by creating multiple copies of the cell at the terminal, each destined to one of the destinations, and routing the copies independently. The number of cell copies may change during the lifetime of a connection depending on the addition and deletion of endpoints over time. Alternatively, multicast routing may be achieved by requiring the switches in the network to have the capability of replicating a cell at several of their output ports according to connection setup information. This mode of operation results in lower traffic congestion in the network at the expense of higher complexity in switch design. Several different approaches for multicasting have been proposed that are suitable for various switch designs. In what follows we discuss four approaches that are shown in Figure 2.15.

Copy while Routing In a multistage virtual circuit switch, where all cells in a virtual circuit follow the same path (i.e. static routing), one can support a multipoint connection by routing the cells through a multicast tree. At each branch in the tree, the cell is sent to specified outputs of the switching element. In point-to-point virtual circuits the input port processor looks up the virtual circuit identifier of the cell, determines a route and places the route in the cell header. For multipoint virtual circuits, a broadcast channel number (BCN) is placed in the header and is used to look up the output ports, stored in the tables at the intermediate switch elements. The lookup tables can be independent of the switches or integrated with them. Each table contains a bit vector associated with a BCN, specifying which outputs of the switch element the cell should be copied to. Finally at the outputs there is a translation from BCN to output VCI. When setting up the multipoint circuit all the tables in the switching elements that are contained in the multicast tree need to be changed.



igure 2.15: Multipoint routing. (a) Copy while routing, (b) Copy then route, (c) Global contention, (d) Recycling.

Copy then Route. Networks with dynamic routing require less memory for a multipoint operation compared to the static routing networks. In such networks a method referred to as copy then route can be used, which is shown in Figure 2.15 (b). For this method two concatenated networks can be used. The first is a copy network and the second a routing network. Virtual circuit translation at input port processors yields a fanout which is used by the copy network to produce an appropriate number of copies. Central routing tables use a broadcast channel number to identify the proper output for each copy and provide the outgoing virtual circuit identifier. Switch elements have no routing tables but make copies based on the fanout and the stage they are in. Because of the dynamic routing, each routing table can potentially receive copies for any connection. However, the copying process is constrained so that each table needs to store information for just one copy from each connection [53].

Global Contention. A contention resolution method for multipoint routing in unbuffered networks is shown in Figure 2.15 (c). First a concentrator puts the cells on consecutive input lines and then an adder computes fanout sums, and marks each cell with the number of copies that are above that particular cell. Then dummy address coders (DAC) filter excess cells that cannot be copied during the cycle, and acknowledge the winners. The coders also compute the output port ranges for each cell, to be used by the copy network. The copy network replicates the cells and labels cells with a copy number. The banyan topology in the copy network ensures that no conflicts occur in the network [38].

Recycling. Multipoint switching can be accomplished by any switching network capable of making two copies of a cell. The approach used is called cell recycling, shown in Figure 2.15 (d). Multicast connections are built as binary trees and the coping is done by passing multiple times through a network, each time making two copies. Virtual circuit translation tables provide information on both copies. The tables store two output port addresses and two virtual circuit identifiers

in addition to a bit specifying if recycling is to take place, for each connection. The network makes a copy of the cells when the routes diverge and then routes each copy to its destination. If a cell is recycled then the new virtual circuit identifier is used for the next table lookup and the process is repeated again. By using recycling, the memory requirements and the connection setup time needed to support multipoint can be reduced to a minimum amount, but the cell delay increases [65].

2.7. Remarks

This chapter gave a general overview of broadband ATM switching. We briefly introduced ATM virtual circuit cell switching and specified the requirements for switching. We then discussed topology and network construction used in multistage switching networks. The different performance measures were defined and an architectural taxonomy presented. The taxonomy is based on network structure, contention resolution and routing strategies used in the systems. Finally we looked at several multipoint approaches that can be used by the various architectures discussed.

3. QUEUEING ANALYSIS OF BUFFERED COPY NETWORKS

An important function in broadband switching systems is support for multipoint connections. One popular paradigm for a multicast switching is concatenation of two networks. In such a cascading, the first network performs cell replication, and is referred to as a copy network, while the second network (called the routing network) performs the routing of cells to the desired outputs. Due to the increasing interest in multipoint networks there is a need to analyze the performance of such networks. The multipoint system under consideration here, called copy-then-route, consists of switching networks that have internal buffering. In this chapter we will focus on the performance of the copy network and present models for the performance evaluation of copy networks. In the next chapter we will consider shared buffered routing networks. Copy networks are mainly used to perform cell replication required for multipoint connections, but are used to distribute cells for load balancing and also to route particular cells. The models presented provide methods for analyzing the queueing behavior of copy networks constructed from binary switches. First we derive a model for switches employing output buffering and then generalize it to include switches with input, and shared buffering. The performance results of the copy networks obtained from the analytical models are compared with simulation results with special attention to their dependence on fanout and network size.

The chapter is organized as follows. First there is a discussion of related work. Then the basic assumptions used in the models and the probabilistic model for various types of single switch elements are presented. In section 3.4. we describe how the model is used to analyze the copy network performance. In Section 3.5. we examine the network performance, and compare the model results with results obtained from simulation.

3.1. Related Work

Performance analysis of switching networks has attracted considerable research interest since the early 1980s. One of the early studies on performance analysis of buffered switching networks was the work of Jenq [32] in 1983. He described a method for analyzing the queueing behavior of binary banyan routing networks with a single buffer at each switching element input. Even though his method does not provide closed form solutions, it does permit efficient computation of the delay and throughput characteristics of a switching network. However in his analysis, Jenq made a number of simplifying assumptions and as a result, Jenq's method provides a simple approximate analysis of buffered switching networks.

Szymanski and Shaikh [54] extended Jenq's method in 1989 in order to analyze switching systems constructed from switches with an arbitrary number of inputs and buffer slots. The buffering options were also extended to include output buffering and a combination of input and output buffering. However, the independence assumption between buffers made in this case yields inaccuracies for large switch and buffer sizes.

Turner [60] extended Szymanski and Shaikh's work to cover switching systems in which the buffer slots in a switch are shared among all the inputs and outputs. Such systems require an analysis which explicitly models the state of the entire switch as opposed to the state of a single buffer in a switch. He also applied the method to systems using parallel bypass input buffering and to the system studied previously by Szymanski and Shaikh. With this method, Turner was able to obtain more accurate results. Later Bianchi and Turner [8] improved upon the analysis for shared buffering with a more accurate model.

Recently Pattavina and Monterosso [48] proposed a method that uses an exact model of a single switch. It gives the most accurate results, but because of its high computational complexity it is only suitable for small switches and small buffer sizes.

All the models mentioned above have emphasized routing networks with uniform traffic. They also make the assumption that switches in different stages in the network are independent. This assumption causes some inaccuracies especially for small switching elements. So far there has little work been done to extend these models to cover networks that do copying of cells and distribution. We have extended previous work on routing networks to cover switching systems which support broadcast capabilities by using binary switch elements.

3.2. Model Assumptions

The results presented here were obtained for a binary delta network but are applicable to any topologically equivalent network such as the banyan and omega networks. The analysis can easily be extended to other network topologies as well. A delta network, $D_{n,d}$ is constructed recursively from d x d switches as shown in Figure 2.7. Such networks provide a single path between any input and output, and have $k = \log_d n$ stages of switching. Multicast networks can be obtained by cascading two such networks together, with the first network (copy network) performing the cell replication, with the second one (routing network) performing the routing of cells to the desired outputs [13].

The switching system is operated in a time-slotted fashion, with fixed length cells progressing synchronously from stage to stage. Low level flow control mechanisms regulate the flow of cells between stages, to prevent cells from being lost due to buffer overflow. This is accomplished by granting permission to the upstream neighbor to send a cell when there is enough buffer space for a cell to be received in the switch element. All cell arrivals are assumed to be Bernoulli distributed, as commonly used in such analyses, even though this assumption is not strictly true. An arriving cell has a fanout associated with it corresponding to the number of copies to be made in the copy network. Cells are replicated in the network according to their fanout. The fanout value for cells at

each input is assumed to be independent from fanout values of cells at other inputs and follows some arbitrary distribution.

3.3. Model of Copy Switch Elements

Before we can analyze the queueing behavior of a buffered copy network, we need to model the switch elements. We will look at four different switch elements: Output buffered, shared buffered with virtual output buffers, shared buffered, and input buffered. The basic buffering schemes are shown in Figure 3.1. In an output buffer switch element, both inputs have access to the two buffers. For this configuration copy cells need to be copied upon arrival and output ports have to be selected for distribution cells. In a shared buffer switch element all the buffer slots have access to the inputs and outputs. Distribution cells can be assigned to an output port upon arrival (virtual output buffer), or sent to an available output port at departure (shared buffer). Finally an input buffer switch element has FIFO input buffers at both of the inputs, requiring the cells at the front of the buffers to compete for the output ports at each cycle.

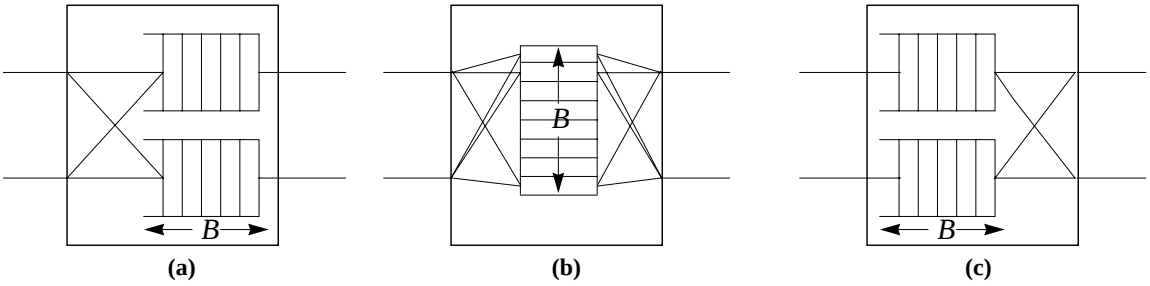


Figure 3.1: Switch Elements: Output buffer (a), shared buffer (b) and input buffer (c). *B* is the number of buffer slots per buffer.

All the switch elements can be modelled similarly with a Markov chain. In order to illustrate the model we will first look at the output buffer switch element. We will then generalize the model and apply it to the other types of switch elements.

3.3.1. Output Buffer Model

The state of the switch can be represented by a vector $\mathbf{s} = [s_0, s_1]$, $0 \le s_0, s_1 \le B$, where s_0 represents the number of cells in output buffer 0, s_1 represents the number of cells in output buffer 1, and B is the size of the buffers. Let $\pi(\mathbf{s})$ be the *steady state probability* that a switch is in state $\mathbf{s}$. The probability that a switch leaves state $\mathbf{s}_1$ and goes to state $\mathbf{s}_2$ is denoted by the *transition probabilities* $\lambda(\mathbf{s}_1, \mathbf{s}_2)$. Then the steady state probability that a switch is in state $\mathbf{s}_2$ can be expressed in terms of the transient probabilities and the steady state probabilities of the switch being in state $\mathbf{s}_1$:

$$\pi(\mathbf{s}_2) = \sum_{\forall \mathbf{s}_1} \pi(\mathbf{s}_1)\lambda(\mathbf{s}_1, \mathbf{s}_2) \quad . \tag{1}$$

The transition from state $\mathbf{s}_1$ to state $\mathbf{s}_2$ can be divided into two independent parts. The first part of the transition is due to cell arrival, assuming no departures of cells have occurred, and the second part is due to cell departure, assuming no cell arrival has occurred. Let the *arrival probability*, i.e. the probability that cells arrive at the switch cause a transition of the switch state from state $\mathbf{s}$ to state $\mathbf{k}+\mathbf{s}$, assuming no departure, be given by $p(\mathbf{k}|\mathbf{s})$, where $\mathbf{k}$ is a vector representing the cell arrivals. Let the *departure probability*, i.e. the probability that cells departing cause a transition from state $\mathbf{s}$ to state $\mathbf{s}-\mathbf{h}$, assuming no arrival, be denoted by $q(\mathbf{h}|\mathbf{s})$, where $\mathbf{h}$ is a vector representing the cell departures. The transition probabilities can then be obtained as follows:

$$\lambda(\mathbf{s}_1, \mathbf{s}_2) = \sum_{\forall \mathbf{k}, \mathbf{h} \in R} p(\mathbf{k}|\mathbf{s}_1)q(\mathbf{h}|\mathbf{s}_1) \qquad \text{where} \qquad R = \{\mathbf{k}, \mathbf{h} | \mathbf{s}_2 = \mathbf{s}_1 + \mathbf{k} - \mathbf{h}\} \tag{2}$$

Thus in order to obtain $\lambda(\mathbf{s}_1, \mathbf{s}_2)$ and $\pi(\mathbf{s})$ one needs to evaluate $p(\mathbf{k}|\mathbf{s})$ and $q(\mathbf{h}|\mathbf{s})$. The probability of arrival depends both on the probability that cells are available at the inputs and the probability that a switch can accept the cells. We first find the probability of accepting a cell i.e

that a switch gives a grant, which only depends on the occupancy of the buffers and then condition the arrivals on the grants.

Let the grant flow control be specified by the grant vector $\mathbf{z} = [z_0, z_1], z_0, z_1 \in \{0, 1\}$ where value 0 for z_m corresponds to no grant, and 1 corresponds to a grant for port m. Let $G(\mathbf{z}|\mathbf{s})$ be the probability that a switch gives a grant to inputs according to the grant vector $\mathbf{z}$ when in state $\mathbf{s}$. The output buffer switch element gives grants to both inputs when there are at least two buffer slots available in both buffers and to one input if there is at least one slot available. Then:

$$G(\mathbf{z}|\mathbf{s}) = \begin{cases} 1 & (\mathbf{z} = [0, 0] \wedge \max(s_0, s_1) = B) \vee (\mathbf{z} = [1, 1] \wedge \max(s_0, s_1) \leq B - 2) \\ \frac{1}{2} & (\mathbf{z} = [1, 0] \vee \mathbf{z} = [0, 1]) \wedge \max(s_0, s_1) = B - 1 \\ 0 & \text{otherwise.} \end{cases}$$

Let α_m be the probability that a cell is available to enter on an input port $m \in \{0, 1\}$ of a switch, and let β_m be the probability that an output port receives a grant to transmit a cell at the output. Furthermore we assume there are four kinds of cells arriving at an input of a switch element: cells requesting output port 0, cells requesting output port 1, cells that need to be copied to both output ports (copy cells), and cells that can be distributed to either output port (distribution cells). Let $r_{m,0}$, $r_{m,1}$, $r_{m,2}$, and $r_{m,3}$ be the probabilities of the different types of cells for an input m respectively.

We model the potential impact of arrivals on input m, when the switch is in state $\mathbf{s}$ with a generating function $A(m|\mathbf{s})$ on the variables x_0 and x_1 where the coefficient of $x_0^i x_1^j$ is the probability that cell arrivals at input m add i cells to output buffer 0 and j cells to output buffer 1. $A(m|\mathbf{s})$ is given by:

$$A(m|\mathbf{s}) = 1 - \alpha_m + \alpha_m (0.5 r_{m,3} + r_{m,0}) x_0 + \alpha_m (0.5 r_{m,3} + r_{m,1}) x_1 + \alpha_m r_{m,2} x_0 x_1 \quad \forall \mathbf{s}$$

Note that the distribution cells are assigned uniformly to an output buffer. when the celll arrives to the switching element.

Next we combine the arrivals on both of the input ports and condition the arrivals on the grant vector by letting $C(\mathbf{k}|\mathbf{z}, \mathbf{s})$ be the coefficient of the term $x_0^{k_0} x_1^{k_1}$ in the product $\prod_{\forall m,\, z_m = 1} A(m|\mathbf{s})$ where $\mathbf{k} = [k_0, k_1]$. Then the arrival probability is just the sum over all the possible grant vectors of the predefined probabilities, and is given by:

$$p(\mathbf{k}|\mathbf{s}) = \sum_{\forall \mathbf{z}} G(\mathbf{z}|\mathbf{s}) C(\mathbf{k}|\mathbf{z}, \mathbf{s}) \quad . \tag{3}$$

To facilitate the evaluation of $q(\mathbf{h}|\mathbf{s})$ we define the conditional probability $q(\mathbf{h}|\mathbf{z}, \mathbf{s})$, which is the probability that the departure of cells causes a state transition from state $\mathbf{s}$ to state $\mathbf{s}$-$\mathbf{h}$ given grants from the outputs that correspond to grant vector $\mathbf{z}$, as:

$$q(\mathbf{h}|\mathbf{z}, \mathbf{s}) = \begin{cases} 1 & h_i = \min(s_i, z_i) \qquad i \in \{0, 1\} \\ 0 & \text{otherwise} \end{cases}$$

Then the departure probability is the sum of all the possible grant vectors of the conditional departure probability weighted by the associated grant probability, and is given by:

$$q(\mathbf{h}|\mathbf{s}) = \sum_{\forall \mathbf{z}} \left[\left(\prod_{\forall m,\, z_m = 1} \beta_m \right) \left(\prod_{\forall m,\, z_m = 0} (1 - \beta_m) \right) q(\mathbf{h}|\mathbf{z}, \mathbf{s}) \right] \quad . \tag{4}$$

Other probabilities needed for the copy network analysis including the *potential throughput* and the *grant probability,* can be obtained from the steady state probability. The grant probability, g_m, is the probability that a switch gives a grant to input *m*, signifying that the switch can accept a cell that arrives on the input in a given cycle, and is given by:

$$g_m = \sum_{\forall \mathbf{s}} \sum_{\forall \mathbf{z},\, z_m = 1} G(\mathbf{z}|\mathbf{s}) \pi(\mathbf{s}) \qquad \text{where} \qquad \mathbf{z} = [z_0, z_1] \wedge z_m = 1 \,. \tag{5}$$

The potential throughput is the probability that a cell is available to leave from output *m* of a switch in a given cycle. For the output switch the potential throughput, t_m, is the sum of the steady state probabilities for which the switch has cells for output *m* and it is given by:

$$t_m = \sum_{\forall \mathbf{s}} \pi(\mathbf{s}) \qquad \text{where} \qquad \mathbf{s} = [s_0, s_1] \wedge s_m > 0$$

The above analysis is a special case of a general form that can be used for other types of switch elements as well. In general, the state $\mathbf{s}$ is a vector $[s_0, s_1, s_2, \ldots, s_{r-1}]$ with *r* elements. $\mathbf{h}$, and $\mathbf{k}$ are similar vectors representing state changes. Then $\pi(\mathbf{s})$, $\lambda(\mathbf{s}_1, \mathbf{s}_2)$, $p(\mathbf{k}|\mathbf{s})$, $q(\mathbf{h}|\mathbf{s})$, and g_m, are defined as before and equations (1)-(5) can be generalized accordingly and then can be applied directly. We proceed now with the generalization of the model. We define accordingly $A(m|\mathbf{s})$ as the generating function, where now the coefficient of the term $x_0^{k_o} \ldots x_{r-1}^{k_{r-1}}$ is the probability that cell arrivals at input *m*, add k_i to component s_i of the state vector. $C(\mathbf{k}|\mathbf{z}, \mathbf{s})$ is the coefficient of $x_0^{k_0} x_1^{k_1} \ldots x_{r-1}^{k_{r-1}}$ in $\prod_{\forall m,\, z_m = 1} A(m|\mathbf{s})$ where now $\mathbf{k} = [k_0, k_1, \ldots, k_{r-1}]$. To define a new switch element, we must define the state of the switch and show how to compute $G(\mathbf{z}|\mathbf{s})$, $A(m|\mathbf{s})$, $q(\mathbf{h}|\mathbf{z}, \mathbf{s})$, and t_m. Everything else has been defined and can now be applied to the other switch element types.Table 3.1 summarizes the different notation and symbols used.

Table 3.1: Summary of notation

Symbol	Comments
B	Number of buffer slots per buffer.
α_m	Probability that a cell is available to enter at an input *m*.
β_m	Probability that an output port receives a grant to transmit a cell at output *m*.

Table 3.1: Summary of notation

Symbol	Comments
$r_{m,i}$	Probability that cell arriving on input m is; destined to output port 0 (i = 0), output port 1 (i = 1), need to be copied to both output ports (i = 2), or can be distributed to either output port (i = 3).
$\pi(\mathbf{s})$	Steady state probability that switch is in state $\mathbf{s}$.
$\lambda(\mathbf{s}_1, \mathbf{s}_2)$	Transition probability that switch leaves state $\mathbf{s}_1$ and goes to state $\mathbf{s}_2$.
$p(\mathbf{k}\|\mathbf{s})$	Arrival probability that cells arrive at the switch cause a transition of the switch state form state $\mathbf{s}$ to state $\mathbf{k}+\mathbf{s}$.
$q(\mathbf{h}\|\mathbf{s})$	Departure probability that cells departing the switch cause a transition from state $\mathbf{s}$ to state $\mathbf{s}-\mathbf{h}$.
$G(\mathbf{z}\|\mathbf{s})$	Probability that s switch gives a grant to inputs according to the grant vector $\mathbf{z}$ when in state $\mathbf{s}$.
$A(m\|\mathbf{s})$	Generating function on the variables x_0 and x_1 where the coefficient of $x_0^i x_1^j$ is the probability that cell arrivals at input m add i cells to output buffer 0 and j cells to output buffer 1.
$C(\mathbf{k}\|\mathbf{z}, \mathbf{s})$	
$q(\mathbf{h}\|\mathbf{z}, \mathbf{s})$	Probability that departure of cells causes a state transition from state $\mathbf{s}$ to state $\mathbf{s}-\mathbf{h}$ given grants from the outputs that correspond to grant vector $\mathbf{z}$.
g_m	Probability that a switch gives a grant to input m.
t_m	Probability that a cell is available to leave from output m of a switch.

3.3.2. Shared Virtual Output Buffer Model

The virtual output switch element is a shared buffer switch element in which we assign distribution cells to a particular output port upon arrival of the cells. Copy cells are not copied until departure in order to avoid waste of buffer slots. Such a switch element has a state representation

$$\mathbf{s} = [s_0, s_1, s_2], \quad 0 \le s_2 \le B \ \wedge \ s_2 \le s_0, s_1 \le B \ \wedge \ \sigma = s_0 + s_1 - s_2 \le B$$

where s_0 represents the number of cell copies in the buffer destined to output 0, s_1 the number of cell copies in the buffer destined to output 1, s_2 the number of copy cells that are yet to be copied,

and B is the size of the buffer. In this state representation, cells that have yet to be copied are counted in all three components of the state vector. So for example, the state vector [1, 1, 1] represents a state in which there is a single cell, which is to be sent to both outputs. The distribution cells are assumed to be assigned to a random virtual output buffer randomly with uniform probability.

The probability that a switch in state $\mathbf{s}$ gives a grant to inputs corresponding to grant vector $\mathbf{z}$ depends on the number of available buffer slots. When there are two or more slots available in the buffer, a grant is given to both inputs, when there is one slot available only one input gets a grant, and if the buffer is full neither input gets a grant. Thus the probability $G(\mathbf{z}|\mathbf{s})$ becomes:

$$G(\mathbf{z}|\mathbf{s}) = \begin{cases} 1 & (\mathbf{z} = [0, 0] \wedge B - \sigma = 0) \vee (\mathbf{z} = [1, 1] \wedge B - \sigma \geq 2) \\ 1/2 & (\mathbf{z} = [1, 0] \vee \mathbf{z} = [0, 1]) \wedge B - \sigma = 1 \\ 0 & \text{otherwise} \end{cases}$$

The generating function that represents the potential impact of arrivals on input m, when in state $\mathbf{s}$ remains the same as for the output buffer case, except for the last term, and is given by:

$$A(m|\mathbf{s}) = 1 - \alpha_m + \alpha_m (0.5 r_{m,3} + r_{m,0}) x_0 + \alpha_m (0.5 r_{m,3} + r_{m,1}) x_1 + \alpha_m r_{m,2} x_0 x_1 x_2 \qquad \forall \mathbf{s}$$

Recall that we need to determine $q(\mathbf{h}|\mathbf{z}, \mathbf{s})$, the probability that the departure of cells causes a state transition from state $\mathbf{s}$ to state $\mathbf{s}$-$\mathbf{h}$ given the grants from outputs corresponding to $\mathbf{z}$. First we need to consider what cell to select when a given output has cells of more than one type destined for it. The issue here is how should the choice be made given that our state representation does not include time ordering information on the cells for a given output, and thus some other criterion instead of time must be used. We will adopt the approach of selecting the cell type based on how many of the total cells associated with that output belong to each type. Thus, the probability $q(\mathbf{h}|\mathbf{z}, \mathbf{s})$ is given by:

$$q(\mathbf{h}|\mathbf{z}, \mathbf{s}) = \begin{cases} 1 & |\mathbf{h}| = 0 \wedge (|\mathbf{z}| = 0 \vee ((\forall j \ \min(z_j, s_j) = 0) \wedge s_2 = 0)) \\ (s_i - s_2)/s_i & |\mathbf{h}| = 1 \wedge h_i = 1 \wedge (\forall j \ h_j = \min(z_j, s_j)) \\ s_2/s_i & |\mathbf{h}| = 2 \wedge h_i = h_2 = 1 \wedge (\forall j \ h_j = \min(z_j, s_j)) \\ \dfrac{(s_0 - s_2)(s_1 - s_2)}{s_0 s_1} & |\mathbf{h}| = 2 \wedge (\forall j \ h_j = \min(z_j, s_j) = 1) \\ 1 - \dfrac{(s_0 - s_2)(s_1 - s_2)}{s_0 s_1} & |\mathbf{h}| = 3 \wedge (\forall j \ h_j = \min(z_j, s_j) = 1) \\ 0 & \text{otherwise} \end{cases}$$

where $i, j \in \{0, 1\}$ and $|\cdot|$ is the sum of the components of a vector. The first condition corresponds to the case when no cell leaves, the second condition corresponds one cell destined for a particular output leaving. The third case represents a cell copied to one of the outputs, the fourth case corresponds to two cells leaving and neither is a copy, and the last one corresponds to two cells leaving and at least one of them is a copy. Note that when two copies leave the switch element we assume that they are from the same original copy cell, thus causing the number of copy cells to decrease.

The probability that a cell is available to leave on output *m* of a switch in a given cycle is the sum of the steady state probabilities for which the switch has cells for output *m*. This contains all states in which the switch has at least one cell destined for output *m* or it has at least one copy cell. Thus the potential throughput is given in this case by:

$$t_m = \sum_{\forall \mathbf{s} \in R} \pi(\mathbf{s}) \qquad \text{where} \qquad R = \{ [s_0, s_1, s_2] \, | s_m \neq 0 \}$$

3.3.3. Shared Buffer Model

In a shared buffer switch element the outputs share the buffer space. The copy cells are copied at departure, and distribution cells are routed to either output available at departure. Such a switch element has a state representation

$$\mathbf{s} = [s_0, s_1, s_2, s_3], 0 \leq s_2, s_3 \leq B \wedge s_2 \leq s_0, s_1 \leq B \wedge \sigma = s_0 + s_1 - s_2 + s_3 \leq B$$

where s_1 represents the number of cell copies in the buffer destined to output 0, s_1 the number of cell copies in the buffer destined to output 1, s_2 the number of copy cells in the buffer that are yet to be copied, s_3 the number of distribution cells in the buffer, and B is the size of the buffer. Note that the copy cells are counted by s_2 but also taken into account in s_0, and s_1.

The probability that a switch in state $\mathbf{s}$ gives a grant to inputs corresponding to grant vector $\mathbf{z}$ is the same as for the shared virtual output buffer and is given by:

$$G(\mathbf{z}|\mathbf{s}) = \begin{cases} 1 & (\mathbf{z} = [0,0] \wedge B - \sigma = 0) \vee (\mathbf{z} = [1,1] \wedge B - \sigma \geq 2) \\ 1/2 & (\mathbf{z} = [1,0] \vee \mathbf{z} = [0,1]) \wedge B - \sigma = 1 \\ 0 & \text{otherwise} \end{cases}$$

The generating function that represents the potential impact of arrivals on input m, when in state $\mathbf{s}$, differs from the shared virtual output buffer since the distribution cells are included in the state representation. The function in this case is given by:

$$A(m|\mathbf{s}) = 1 - \alpha_m + \alpha_m r_{m,0} x_0 + \alpha_m r_{m,1} x_1 + \alpha_m r_{m,2} x_0 x_1 x_2 + \alpha_m r_{m,3} x_3 \qquad \forall \mathbf{s}$$

To calculate the departure probabilities we use the same criterion for the selection of the type of cell as in the shared virtual output buffer case. However, in this case we have to consider how to deal with the distribution cells. We adopt the position that distribution cells can contend for both output ports. With this assumption the departure probabilities for each port are straightforward and given by:

$$q(\mathbf{h}|\mathbf{z},\mathbf{s}) = \begin{cases} 1 & |\mathbf{h}| = 0 \wedge (|\mathbf{z}| = 0 \vee ((\forall i \ \min(z_i, s_i) = 0) \wedge s_2 = s_3 = 0)) \\ \dfrac{(s_i - s_2)}{(s_i + s_3)} & |\mathbf{h}| = 1 \wedge h_i = 1 \wedge \min(z_i, s_i) = 1 \wedge \min(z_{1-i}, s_{1-i} + s_3) = 0 \\ \dfrac{s_2}{(s_i + s_3)} & |\mathbf{h}| = 1 \wedge h_i = h_2 = 1 \wedge \min(z_i, s_i) = 1 \wedge \min(z_{1-i}, s_{1-i} + s_3) = 0 \\ \dfrac{(s_0 - s_2)\ (s_1 - s_2)}{(s_0 + s_3)\ (s_1 + s_3)} & |\mathbf{h}| = 2 \wedge (\forall i \ h_i = \min(z_i, s_i) = 1) \\ \dfrac{(s_0 + s_1 - s_2 + s_3)\, s_2}{(s_0 + s_3)\ (s_1 + s_3)} & |\mathbf{h}| = 3 \wedge h_3 = 0 \wedge (\forall i \ h_i = \min(z_i, s_i) = 1) \\ Q(\mathbf{z},\mathbf{s}) & \mathbf{h} = [0, 0, 0, 1] \\ \sum\limits_{\mathbf{u},\mathbf{v} \in D(\mathbf{h})} \frac{1}{2}[q(\mathbf{u}|[1,0],\mathbf{s})q(\mathbf{v}|[0,1],\mathbf{s}-\mathbf{u}) + q(\mathbf{u}|[1,0],\mathbf{s}-\mathbf{v})q(\mathbf{v}|[0,1],\mathbf{s})] & \\ & |\mathbf{h}| = 2 \wedge h_3 > 0 \wedge |\mathbf{z}| = 2 \\ \dfrac{1}{2}\dfrac{s_2 s_3}{(s_i + s_3 - 1)\ (s_{1-i} + s_3)} & |\mathbf{h}| = 3 \wedge h_i = h_2 = h_3 = 1 \wedge s_3 > 0 \wedge |\mathbf{z}| = 2 \\ 0 & \text{otherwise} \end{cases}$$

where $i \in \{0, 1\}$, $D(\mathbf{h})$ is the set of all pairs $(\mathbf{u}, \mathbf{v})$ with $\mathbf{h} = \mathbf{u} + \mathbf{v}$ and $u_1 = v_0 = 0$ and

$$Q(\mathbf{z},\mathbf{s}) = \begin{cases} 1 & \mathbf{z} = [1, 1] \wedge \mathbf{s} = [0, 0, 0, 1] \\ s_3 / (s_i + s_3) & |\mathbf{z}| = 1 \wedge s_3 > 0, i \in \{0, 1\} \\ s_3 / 2\,(s_i + s_3) & \mathbf{z} = [1, 1] \wedge s_i > 0 \wedge s_{1-i} = 0 \wedge s_3 = 1, \quad i \in \{0, 1\} \\ 0 & \text{otherwise} \end{cases}$$

The probability that a cell is available to leave on output *m* of a switch in a given cycle is the sum of the steady state probabilities for which the switch has a cell for output $m \in \{0, 1\}$. Let the set of states $\mathbf{s} = [s_0, s_1, s_2, s_3]$ for which $s_m > 0 \vee s_3 > 1$ be represented by the set R_1 and the set of states for which $s_m = 0 \wedge (s_{1-m} > 0) \wedge s_2 = 0 \wedge s_3 = 1$ be represented by R_2.

Then:

$$t_m = \sum_{\mathbf{s} \in R_1} \pi(\mathbf{s}) + \sum_{\mathbf{s} \in R_2} x\pi(\mathbf{s}) + (0.5\,(\beta_0 + \beta_1 - \beta_0\beta_1) \,/\, \beta_m)\,\pi([0, 0, 0, 1])$$

where $x = [\beta_0\beta_1\,(s_m + 0.5) \,/\, (s_m + 1) + (1 - \beta_m)\,\beta_{1-m}]\,\dfrac{1}{\beta_m}$

3.3.4. Input Buffer Model

In the input buffer switch element the buffer is located at the input ports and the cells at the front of buffers are considered for departure at each cycle. It has a state representation

$$\mathbf{s} = [s_0, s_1, s_2, s_3], \quad 0 \le s_0, s_1 \le B, \quad s_2, s_3 \in \{0, 1, 2, 3, 4\}$$

where s_0 represents the number of cells in input buffer 0 and s_1 the number of cells in input buffer 1. s_2 and s_3 specify the type of the first cell in buffer 0 and 1 respectively, where there are five types corresponding to no cell, cell destined to output 0, cell destined to output 1, copy cell and distribution cell, denoted as 0, 1, 2, 3, and 4 respectively. B is the size of each buffer.

The input buffer model differs from the general case in that the departures and arrivals are not completely independent. This is because when the last cell departs from a buffer the cell type in the front of the buffers, denoted by s_2 and s_3 change. It is therefore necessary to modify the transition probabilities in Equation (2) as shown:

$$\lambda(\mathbf{s}, \mathbf{t}) = \sum_{\forall \mathbf{h}, \mathbf{k}} q(\mathbf{h}|\mathbf{s})p(\mathbf{k}|\mathbf{u}) \qquad \begin{array}{c}\text{where} \\ \text{and}\end{array} \quad \begin{array}{l}\mathbf{t} = \mathbf{s} + \mathbf{k} - \mathbf{h} \\ \mathbf{u} = [s_0, s_1, s_2 - h_2, s_3 - h_3]\end{array} \tag{6}$$

The probability that a switch in state $\mathbf{s}$ gives a grant to inputs corresponding to the vector $\mathbf{z} = \{[z_0, z_1] \,|\, z_0, z_1 \in \{0, 1\}\}$ depends on if the particular input buffer is full or not and is therefore given by:

$$G(\mathbf{z}|\mathbf{s}) = \begin{cases} 1 & (\mathbf{z} = [0,0] \wedge s_0 = B \wedge s_1 = B) \\ 1 & (\mathbf{z} = [1,0] \wedge s_0 < B \wedge s_1 = B) \\ 1 & (\mathbf{z} = [0,1] \wedge s_0 = B \wedge s_1 < B) \\ 1 & (\mathbf{z} = [1,1] \wedge s_0 < B \wedge s_1 < B) \\ 0 & \text{otherwise} \end{cases}$$

The generating function that represents the potential impact of arrivals on input m, when in state $\mathbf{s}$, is given by:

$$A(m|\mathbf{s}) = \begin{cases} (1-\alpha_m) + \alpha_m x_m & s_m > 0 \\ (1-\alpha_m) + \sum_{i=0}^{3} \alpha_m r_{m,i} x_m x_{m+2}^{i+1} & s_m = 0 \end{cases}$$

When the buffer is not empty the constant term represents the probability that no cell arrives and the coefficient of x_m represents the probability that a cell arrives. The type of cell does only matter when the buffer is empty. Then the coefficient of $x_m x_{m+2}^{i+1}$ represents the probability that cell of type i+1 arrives.

Recall that $q(\mathbf{h}|\mathbf{z}, \mathbf{s})$ is the probability that the departure of cells causes a state transition from state $\mathbf{s}$ to state $\mathbf{s}$-$\mathbf{h}$ given the grants from outputs corresponding to $\mathbf{z}$. Earlier we had defined this probability in terms of departure from each of the outputs. However, for the input buffer we need to define it in terms of departure from each of the input buffers. The departure probability can be decomposed by considering each input buffer separately and keeping track of available output ports. Assuming that the buffers are selected randomly with equal probability, we have:

$$q(\mathbf{h}|\mathbf{z}, \mathbf{s}) = 0.5 \sum_{\substack{\forall \mathbf{z}', \mathbf{z}'' \\ z''_i \le z'_i \le z_i}} [\, U_0(h_0, h_2, \mathbf{z}'|\mathbf{z}, s_0, s_2) U_1(h_1, h_3, \mathbf{z}''|\mathbf{z}', s_1, s_3) + U_0(h_0, h_2, \mathbf{z}''|\mathbf{z}', s_0, s_2) U_1(h_1, h_3, \mathbf{z}'|\mathbf{z}, s_1, s_3) \,]$$

where $U_i(\cdot)$ is a probability distribution that represents the departure from an input buffer i which

has s_i cells and a first cell of type s_{i+2} and available output ports $\mathbf{z}$. After departure the buffer will have $s_i - u_i$ cells and cell type $s_{i+2} - u_{i+2}$ and the departure will result in available output ports $\mathbf{z}'$. The departure can be divided in two stages. First we consider the resulting cell type after cell departure or copying from the buffers, but before considering the next cell in the buffer. If a cell departs, the intermediate cell type is set to 0. We then reduce the cell count in the buffer and assign a type to the new cell. The departure probability is then the sum of all the intermediate states:

$$U_m(u_0, u_1, \mathbf{z}' \,|\, \mathbf{z}, s_0, s_1) = \sum_{\substack{\forall v, w \\ u_1 = s_1 - v + w}} V(w, \mathbf{z}' \,|\, \mathbf{z}, s_1) W_m(u_0, v \,|\, s_0, s_1 - w)$$

where $V(\cdot)$ gives the intermediate cell type is given by

$$V(w, \mathbf{z}' \,|\, \mathbf{z}, s) = \begin{cases} 1 & u = 0 \wedge [\,(\mathbf{z} = \mathbf{z}' = [0,0]) \vee (s = 0 \wedge \mathbf{z} = \mathbf{z}')\,] \\ & \quad \wedge [\,(s = 1 \wedge \mathbf{z} = \mathbf{z}' = [0,1]) \vee (s = 2 \wedge \mathbf{z} = \mathbf{z}' = [1,0])\,] \\ 1 & u = 1 \wedge [\,(s = 1 \vee s = 3) \wedge (\mathbf{z} = [1,0] \wedge \mathbf{z}' = [0,0])\,] \\ & \quad \wedge [s = 3 \wedge \mathbf{z} = [1,1] \wedge \mathbf{z}' = [0,1]\,] \\ 1 & u = 2 \wedge [\,(s = 2 \vee s = 3) \wedge (\mathbf{z} = [0,1] \wedge \mathbf{z}' = [0,0])\,] \\ & \quad \wedge (s = 3 \wedge \mathbf{z} = [1,1] \wedge \mathbf{z}' = [1,0]) \\ 1 & u = 3 \wedge s = 3 \wedge \mathbf{z} = [1,1] \wedge \mathbf{z}' = [0,0] \\ 1 & u = 4 \wedge s = 4 \wedge (\mathbf{z} = [1,0] \wedge \mathbf{z}' = [0,0] \vee \mathbf{z} = [0,1] \wedge \mathbf{z}' = [0,0]) \\ \frac{1}{2} & u = 4 \wedge s = 4 \wedge \mathbf{z} = [1,1] \wedge (\mathbf{z}' = [1,0] \vee \mathbf{z}' = [0,1]) \\ 0 & \text{otherwise} \end{cases}$$

and $W_{\mathrm{m}}(\cdot)$ gives the resulting cell count and cell type:

$$W_m(u_0, u_1 \,|\, s_0, s_1) = \begin{cases} r_{m, u_1 + 1} & s_0 > 1 \wedge s_1 = 0 \wedge u_0 = 0 \wedge u_1 > 0 \\ 1 & s_0 = 1 \wedge s_1 = 0 \wedge u_0 = 1 \wedge u_1 = 0 \\ 1 & \neg(s_0 \geq 1 \wedge s_1 = 0) \wedge u_0 = 0 \wedge u_1 = 0 \\ 0 & \text{otherwise} \end{cases}$$

What remains to complete the analysis is to obtain t_m for the input case. The probability that a cell is available to leave on output *m* of a switch in a given cycle is the sum of the steady state probabilities for which the switch has a cell for output *m*. Let the set of states $\mathbf{s} = [s_0, s_1, s_2, s_3]$ for which $s_2 = m + 1 \vee s_3 = m + 1 \vee s_2 = 3 \vee s_3 = 3 \vee (s_2 = 4 \wedge s_3 = 4)$ be represented by the set R_0, and the set of states for which $s_2 = 2 - m \wedge s_3 = 4$ or $s_2 = 4 \wedge s_3 = 2 - m$ be represented by R_1 and the states where $s_2 = 0 \wedge s_3 = 4$ or $s_2 = 4 \wedge s_3 = 0$ be represented by R_2. Then we can write:

$$t_m = \sum_{\mathbf{s} \in R_0} \pi(\mathbf{s}) + \sum_{\mathbf{s} \in R_1} \frac{1}{\beta_m} (\beta_m - 0.25\beta_0\beta_1)\, \pi(\mathbf{s}) + \sum_{\mathbf{s} \in R_2} \frac{0.5}{\beta_m} (\beta_0 + \beta_1 - \beta_0\beta_1)\, \pi(\mathbf{s})$$

3.4. Analysis of Copy Networks

We now have an explicit model for the state of a single switch which can be used to analyze the queueing behavior of the entire delta network by assuming that the states of the various switches in a stage are independent. Let $\pi_i(\mathbf{s})$ be the state of a switch in stage *i*, $t_{m,i}$ be the potential throughput for output port *m* of a switch in stage *i*, and let $g_{m,i}$ be the acceptance probability for input port *m* of a switch in stage *i*. Let $\alpha_{m,\mathrm{i}}$ be the probability of cell arrival to port *m*, and $\beta_{m,i}$ be the probability of receiving grant for port *m*. If output *n* of switch *j* is connected to input *m* of switch *i* then $\alpha_{m,i} = t_{n,j}$, and similarly if output *m* of switch *i* is connected to input *n* of switch *k* then $\beta_{m,i} = g_{n,k}$. Obviously $\alpha_{m,i}$ and $\beta_{m,i}$ depend on the state probabilities of the neighboring switching elements. An iterative computational method is used in which first arbitrary initial values are assigned to the state probabilities, then values for $\alpha_{m,i}$ and $\beta_{m,i}$ are computed for all stages and finally these values are used to compute new balance equations for the Markov chain from

which the new state probabilities are obtained. The iteration stops when convergence is reached. From experience a solution seems to exist, but it has not been proved nor it is not known if it is unique.

The routing, distribution and copy probabilities are needed for each switching element and are obtained from the input fanout distribution of cells offered to the network. To simplify the evaluation of the fanout distribution we assume that the traffic of the network is uniformly distributed and is the same for all input ports. The stages are numbered from left to right starting with 1. Let $f_i(k)$ be the probability that a cell at stage i has a fanout of k, $0 \le k \le N_i$ where N_i is the maximum fanout for stage i. $f_0(k)$ is the fanout distribution at the input of the network. Then the cell fanout distribution $f_i(k)$ at the output of switch element in stage i is given by

$$f_i(k) = \frac{e_i(k)}{\sum_{n=0}^{N_i} e_i(n)} \qquad \text{where}$$

$$e_i(k) = \begin{cases} 0.5 f_{i-1}(k) & 2k \le N_i \\ 0.5\,(f_{i-1}(k) + f_{i-1}(2k-1)) & 2k = N_i \\ 0.5\,(f_{i-1}(k) + f_{i-1}(2k-1) + 2f_{i-1}(2k) + f_{i-1}(2k+1)) & 2k > N_i \wedge k < N_i \\ 0.5\,(f_{i-1}(k) + 2f_{i-1}(2k) + f_{i-1}(2k+1)) & k = N_i \end{cases}$$

From the fanout distribution at each stage we can find the routing probabilities for stage i, using the fact that the switch elements distribute cells that have fanout of 1 and copy cells that have fanout greater than the reachable number of network outputs from a switch element port. Therefore the following equations specify the routing probabilities:

$$r_{00} = r_{01} = r_{10} = r_{11} = 0.5 f_i(0) \qquad r_{03} = r_{13} = \sum_{k=1}^{N_i} f_i(k) \qquad r_{02} = r_{12} = \sum_{k=N_i}^{N_{i-1}} f_i(k)$$

We now have the routing parameters for all the stages in the network and can apply the iterative algorithm to obtain the performance of a multistage copy network.

3.5. Performance of Copy Networks

In this section we look at the performance of copy networks. First we validate the single 2x2 switch element model developed in the previous section by comparing the achieved throughput obtained from the model to that measured by simulation. The simulations were performed using the general simulation tool for analyzing switching systems which is described in Chapter 5. The tool allows performance evaluation of a variety of different switching architectures. Then we discuss results obtained for multistage copy networks.

3.5.1. Switch element performance

The comparison for each of the four models discussed in the preceding sections with simulation results is performed for various routing, distribution and copying parameters. We have studied five different cases that correspond to the combination of routing parameters shown in the table below, where *exp* denotes the different experiment.

Table 3.2 Routing parameters for experimetns.

exp	$\mathbf{r_{m0}}$	$\mathbf{r_{m1}}$	$\mathbf{r_{m2}}$	$\mathbf{r_{m3}}$
1	0.5	0.5	0.0	0.0
2	0.25	0.25	0.25	0.25
3	0.1	0.1	0.5	0.3
4	0.0	0.0	0.8	0.2
5	0.0	0.0	1.0	0.0

The results from the five experiments are summarized in Figure 3.2, where the graphs plot throughput versus the offered load which is defined as the product of the input load and the average fanout. The results obtained from the analytical model are shown with solid lines and the ones

obtained by simulation are plotted with circles. As can be seen from the graphs there is a good agrement between the model and simulation. This is expected since the analysis closely models the different switches. Furthermore, the models for both output and input buffer switch element are exact for the cell arrival assumption used. However, in the shared buffer and shared virtual output buffer switch elements we make assumptions about the type of cell selected for departure based on the number of cells in the buffer. These assumptions do not affect the shared virtual output buffer, but cause the model for shared buffer switch element to slightly underestimate the throughput for the experiments that have distribution cells. The reason for this is that in the simulation there is cell ordering that causes fewer cells to be considered for departure than in the analysis where all cells are considered. Therefore, in the simulations higher priority gets assigned to old

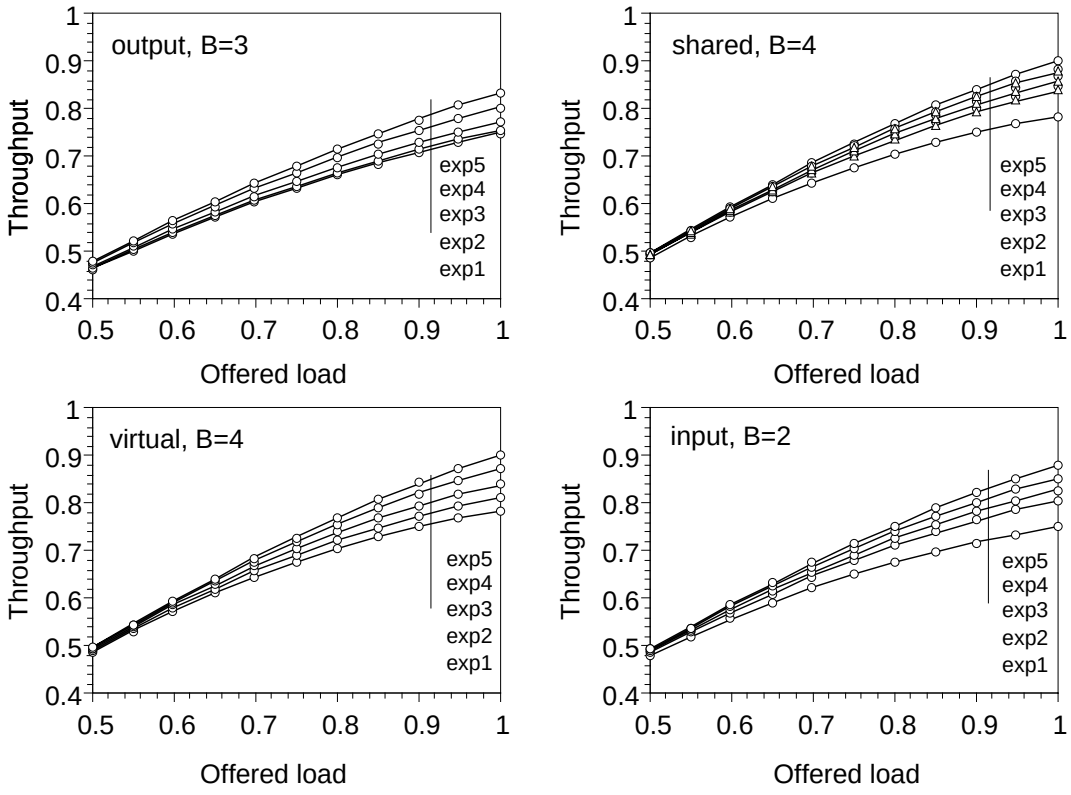


Figure 3.2: Performance comparison of a single switch element. Solid lines refer to analytical model, circles refer to simulation, and triangles to simulations not considering time ordering.

copy cells than to newer distribution cells, resulting in slightly lower throughput. This was verified with more simulations performed by not considering the time information, results of which are plotted in Figure 3.2 for the shared buffer case with triangles. In this case the simulation results match the results from the analytical model.

From the graphs in Figure 3.2 it can be seen that the throughput increases as we increase the copying probability, and that the highest throughput is achieved when all the cells are copy cells, as in *exp5*. On the other hand *exp1* corresponds to a pure routing switch element and has the lowest throughput. All the switch elements have total buffering of two slots per port except for the output switch element. The output switch element has three slots per output port to achieve similar performance as the other switch elements because grant flow control is used.

3.5.2. Performance of multistage copy network

We now examine the performance of a multistage copy network. First we examine the effect of network size on the throughput, then we look at the throughput as a function of offered load for a 16 port network, and finally we examine the dependence of throughput on the fanout. The fanout assigned to each multipoint cell used in the analysis is selected from a truncated geometric distribution with parameter p. More specifically the cell fanout distribution is given by:

$$Pr(\text{fanout} = k) = f_0(k) = \begin{cases} p\,(1-p)^{k-1} & 1 \le k < N \\ (1-p)^{N-1} & k = N \end{cases}$$

where $0 \le p \le 1$, and N is the size of the network. The average fanout obtained with this distribution is given by

$$E(\text{fanout}) = \begin{cases} \left(\frac{1}{p}\right)(1-(1-p)^N) & 0 < p \le 1 \\ N & p = 0 \end{cases}$$

The offered load is defined as $P(\text{cell arrival}) \times E(\text{fanout})$. We also use a deterministic or fixed fanout in the last study. Cells arrive at all the inputs at the same average rate with Bernoulli

distribution. To focus on properties determined by the copy network no backpressure is assumed at the last stage of switching, i.e the last stage always has grant signal asserted. Preceding the copy network we have a input buffer at every input port of size *B*'. In what follows simulation results are plotted with dotted lines but analytical results are plotted with solid lines.

Figure 3.3 shows the effect of the network size on the throughput by summarizing results obtained using the four different types of switches and for two different input buffer sizes. For these experiments the cells have a random fanout distribution with average of 4. It can be seen that in most cases the throughput of a network with no input buffer, $B' = 0$, increases with increasing network size. When the networks have a small input buffer at each input, $B' = 8$, the throughput

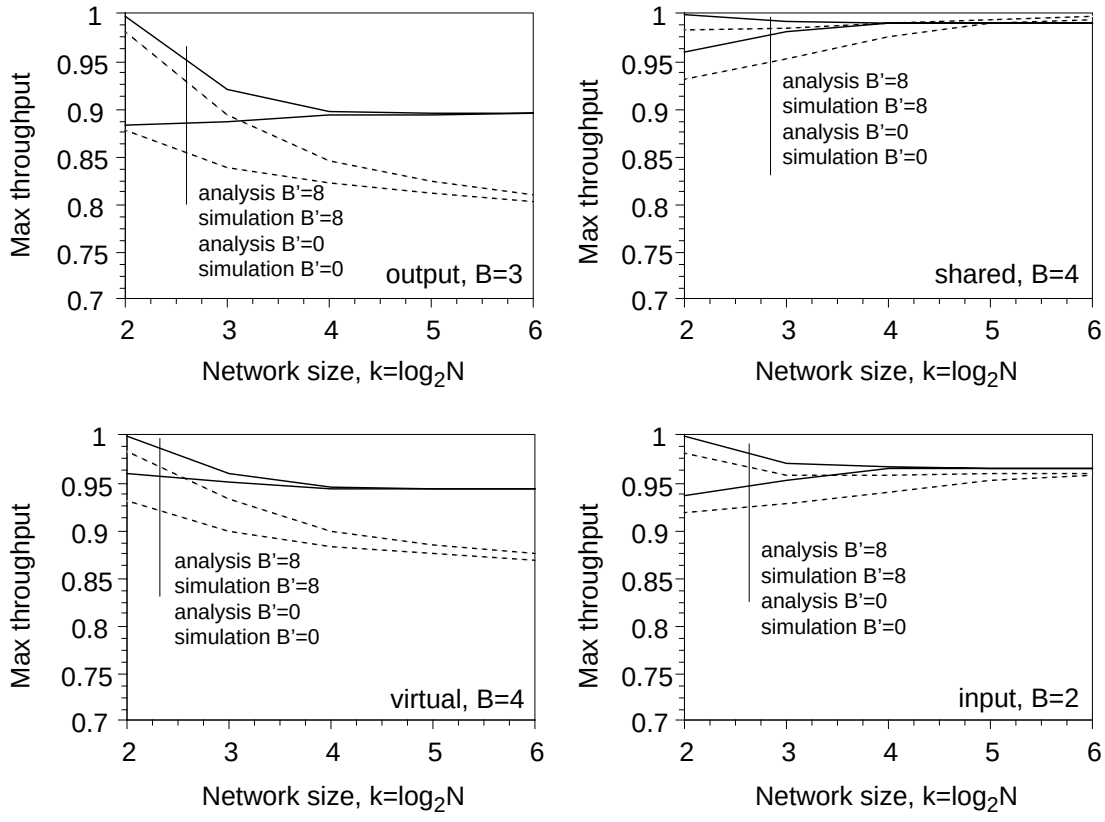


Figure 3.3: Effect of network size on throughput

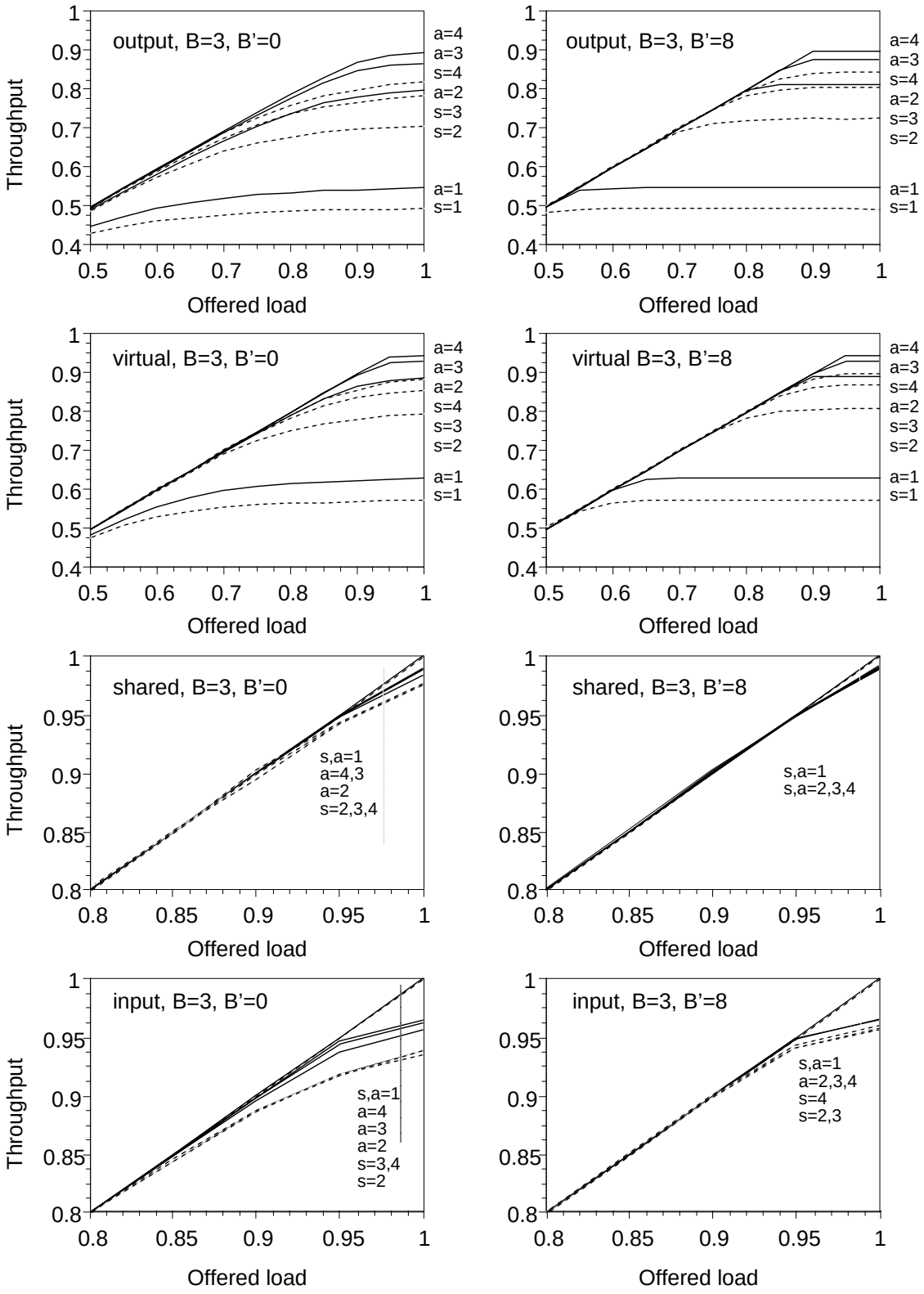


Figure 3.4: Throughput with random fanout for 16 port network. The labels refer to average fanout, a=analytical, s=simulation,

is highest for small networks but decreases to the same level as the no input buffer case. In both cases the throughput levels off to some constant value as the network size increases. Overall the results from the simulation are similar to the analytical results for small networks sizes but deviate as the network size increases especially for output buffering and shared virtual output buffering, which is due to the independence assumption between switch elements in the same stage. The simulations show the same behavior as the analytical models with input buffer networks and networks with no input buffering leveling off to the same throughput when the network size increases.

Next we compare the throughput results from the analytical model and simulation for a 16 port delta network, with and without preceding input buffers. Figure 3.4 shows a set of curves with several cases of geometric fanout labeled with the average fanout value. The plots show the throughput versus the offered load for average fanout of 1,2,3 and 4. For the shared virtual output and the output buffered networks there is an increase in the throughput when the fanout increases. This is because for the lower fanouts it is more likely that cells are to be distributed. Because we make decisions about what output buffers the cells go to when the distribution cells arrive at the switching element the network behaves similar to a routing network and has a lower throughput. As has been seen for analysis of routing networks the throughput obtained from the analytical models is higher than the simulations due to the independence assumptions for the switches in the same network stages.

Overall the curves show that the different fanout distributions have very little effect on the throughput for both the shared and input buffered networks. When all cells are distribution cells (s,a = 1) the network obtains maximum throughput possible since the distribution cells can always be routed to an output port without any conflicts. The throughput is lower for the other fanouts but stays above 95% of the maximum. The simulation results are slightly lower than the analytical results. The set of plots to the right shows the network throughput when the network is preceded

by an 8 slot input buffer. The resulting curves have a sharper knee and slightly higher throughput than the networks without the input buffer.

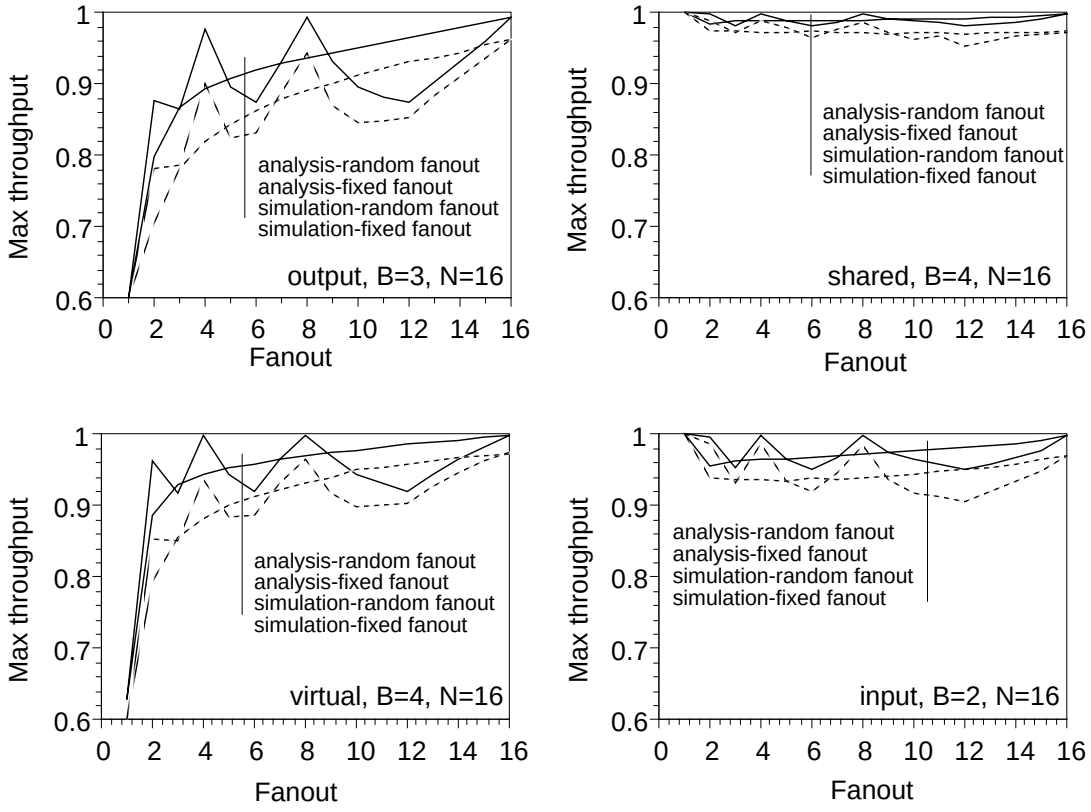


Figure 3.5: Effect of fanout on throughput.

Finally we want compare the resulting throughput predicted by the models with throughput obtained from simulation for different fanout values. Figure 3.5 shows the effect of fanout on the maximum throughput (offered load of 1) for a 16 port copy network. For exponential fanout the throughput increases as the fanout of the cells increases. The reason for the low throughput for small fanout is as mentioned before due to the fact that both types of switch elements make a decision upon the arrival of distribution cells regarding which output buffer to put them in. This reduces the performance when large numbers of cells are to be distributed. The shared buffer and input buffer cases obtain maximum throughput possible for fanout of 1. For fanout of 2 the

throughput drops but then increases slightly as the fanout increases. The jagged curves in all the graphs show the throughput when all cells have the same fixed fanout. The throughput behavior is similar but less drastic to the behavior seen in [13] with the fanouts of powers of two giving the highest throughput, but between the powers of two the throughput drops. This effect is explained by noting that when the fanout passes a power of two, copying begins one stage earlier in the network, while the cell arrival rate is about the same. The reason why the effect is less drastic here is because the copying strategy used is different, allowing partial copying of cells. The results obtained by simulations give a little lower throughput but have the same behavior as the analytical models.

3.6. Remarks

We have developed analytical models for the analysis of copy networks based on input, output and shared buffering. The model is very accurate for a single 2x2 switch element but is approximate for multistage networks. The multistage analytical model overestimates the throughput. The overestimate is largest for output buffer and virtual output buffer due to the independence assumption of network stages. Note that the analytical methods would be more accurate in a multistage case when there is no flow control between stages in the network. The models give a good indication of the dependence of the throughput on the fanout and network sizes. The copy networks can sustain a high throughput in spite of small buffers and act as additional buffering to the routing network. The throughput stays constant or decreases slightly as the network size increases in contrast to the larger decreasing throughput of the routing networks.

The analysis presented here can be easily extended to handle networks that use different flow control methods and non-uniform traffic. Improving the computational performance of these methods would prove to be very useful especially by reducing the total number of iterations needed for

convergence. Extending the models to allow switch elements of arbitrary dimension may prove tractable but the computational complexity is probably too high to be of practical value.

4. QUEUEING ANALYSIS FOR SHARED BUFFER SWITCHING NETWORKS FOR NON-UNIFORM TRAFFIC

Uniform traffic does not necessarily represent a realistic view of traffic patterns in real systems. Non-uniform traffic models better reflect the traffic patterns that need to be accommodated. Non-uniform traffic may cause the network performance to deteriorate to much lower levels than the ones predicted by uniform traffic analysis. Therefore, the performance of buffered networks under non-uniform traffic is an important issue that needs to be studied. In this chapter, we further extend the queueing analysis for buffered networks by providing methods for analyzing the queueing behavior of switching networks under non-uniform traffic patterns. We focus on shared buffer switch elements because they have better performance than input or output buffered elements. The analytical method for performance evaluation is compared with simulation on the basis of accuracy, where the performance is measured in terms of maximum throughput and probability of cell loss.

The chapter is organized as follows. First related work is discussed and then the basic assumptions of the model are presented. Section 4.3 gives a probabilistic model for a single shared buffer switch element. In Section 4.4 we describe how the model is used to analyze the network performance, and in Section 4.5 we examine the network performance and compare the model results with results obtained from simulations.

4.1. Related Work

All of the methods mentioned in Chapter 3 have used the assumption that cell arrivals to the network have destinations which are uniformly distributed among the output ports. This is of course an unrealistic assumption. Wu [69] first presented an analysis of a single buffer binary

banyan network under non-uniform traffic conditions. He proposed an approximate analysis that made some simplifying assumptions. Garg and Huang [26] modified the uniform traffic model proposed by Jenq [32] to study the performance of banyan networks with a single slot input buffer under certain nonuniform traffic patterns. A similar method has been presented by Kim and Leon-Garcia [33] for evaluating the performance of input buffered binary banyan networks operating under non-uniform traffic patterns. In their analysis, each input buffer of a switching element is modeled as a Markov chain, and the relationship between switching elements is described by average flow constraints. They also extended their model to evaluate the performance of multibuffered and parallel banyan networks. Another scheme for performance estimates of single buffered banyan networks for nonuniform traffic has been proposed by Lee [37]. His model achieves more accurate results by including the destination of blocked cells in buffers of nodes at the first stages in his analysis. Similar models have recently been proposed by Atiquzzaman and Akhtar [3] in which they include state information regarding the output link that was requested when a cell was blocked at a switch element. Gianatti and Pattavina [28] studied banyan networks under bursty and unbalanced traffic patterns, but their unbalanced traffic model is simplified and does not allow arbitrary traffic patterns.

All the work mentioned above has shown in general that nonuniform traffic has a detrimental effect on the performance of the network. The non-uniform models that have been proposed are for binary switches with limited numbers of input buffers only, except for the work of Gianatti and Pattavina. We have extended previous work on routing networks to cover non-uniform traffic patterns for networks constructed from switch elements with arbitrary numbers of inputs ports using shared buffer slots.

4.2. Assumptions of Basic Model

The results presented here were obtained using delta networks but are applicable to any topologically equivalent network such as the banyan and omega networks. The analysis can easily be applied to other network topologies as well. The assumptions are similar to those in chapter 3 for the copy networks, with the switching systems operating in a time-slotted fashion, with fixed length cells progressing synchronously from stage to stage. We assume that low level flow control mechanisms regulate the flow of cells between stages to prevent cells from being lost due to buffer overflow. This is accomplished by signaling the upstream neighbor granting permission to send a cell when there is enough buffer space for the cell to be received in the switch element. Consider a shared buffer switch and assume that the number of unoccupied buffer slots is x. The switch grants permission to $\min\{x,d\}$ of its upstream neighbors to send a cell at the start of an operation cycle of the network. If $x < d$, we assume that x predecessors are chosen at random. All cell arrivals are assumed to be Bernoulli distributed, but an arbitrary load matrix representing the traffic pattern can be used.

4.3. Probabilistic Model of Switch Elements

Before we can analyze the queueing behavior of a buffered routing network, we need to model the switch elements. The basic shared buffering scheme is shown in Figure 4.1. In such a switch element the inputs have access to all buffer slots and all the buffer slots have access to the outputs. We model the switch as having virtual output buffers that contain the cells destined for a particular output port The total buffering is limited to B cells.

The switch elements can be modelled with a Markov chain.We will first present an exact model of the switch element, the vector model which is an extension of a uniform model by Pattavina and Monterosso [48] and we use the same notation as in table 3.1. To reduce the complexity we derived an approximate model, which we discuss in Section 4.3.2

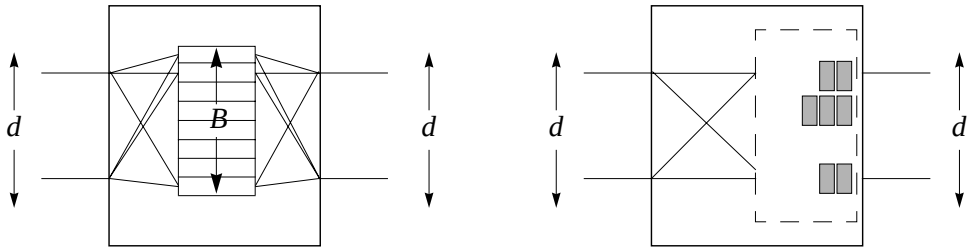


Figure 4.1: Switch Elements: Shared buffer switch element

4.3.1. Vector Model

The state of the switch can be represented by a vector

$$\mathbf{s} = [s_0, s_1, \ldots, s_{d-1}] \qquad \sum_{i=0}^{d-1} s_i \le B \qquad s_i \ge 0, i \in \{0, \ldots, d-1\}$$

where s_i represents the number of cells in the buffer destined for output *i*, *d* is the number of inputs and outputs and *B* is the size of the buffer. Let $\pi(\mathbf{s})$ be the *steady state probability* that a switch is in state $\mathbf{s}$. The probability that a switch leaves state $\mathbf{s}_1$ and goes to state $\mathbf{s}_2$ is denoted by the *transition probabilities* $\lambda(\mathbf{s}_1, \mathbf{s}_2)$. Then the steady state probability can be expressed in terms of the transition probabilities:

$$\pi(\mathbf{s}_2) = \sum_{\forall \mathbf{s}_1} \pi(\mathbf{s}_1)\lambda(\mathbf{s}_1, \mathbf{s}_2) \quad . \tag{7}$$

The transition from state $\mathbf{s}_1$ to state $\mathbf{s}_2$ can be divided in two independent parts. The first part of the transition is due to cell arrival, assuming no departures of cells has occurred, and the second part is due to cell departure, assuming no cell arrival has occurred. Note that the arriving cells do not depart in the same cycle. Let the *arrival probability*, i.e. the probability that cells arriving cause a transition from state $\mathbf{s}$ to state $\mathbf{k}+\mathbf{s}$, assuming no departure, be given by $p(\mathbf{k}|\mathbf{s})$, where

$\mathbf{k}$ is also a d element vector. Let the *departure probability*, i.e. the probability that cells departing cause a transition from state $\mathbf{s}$ to state $\mathbf{s}$-$\mathbf{h}$, assuming no arrival, be denoted by $q(\mathbf{h}|\mathbf{s})$. Once the probabilities $p(\mathbf{k}|\mathbf{s})$ and $q(\mathbf{h}|\mathbf{s})$ are determined then the transition probabilities can be obtained as follows:

$$\lambda(\mathbf{s}_1, \mathbf{s}_2) = \sum_{\forall \mathbf{k}, \mathbf{h} \in R} p(\mathbf{k}|\mathbf{s}_1) q(\mathbf{h}|\mathbf{s}_1) \qquad \text{where} \qquad R = \{\mathbf{k}, \mathbf{h} | \mathbf{s}_2 = \mathbf{s}_1 + \mathbf{k} - \mathbf{h}\} \qquad (8)$$

Hence, we need to determine the departure and the arrival probabilities, but first we need to define several additional probabilities. The probability of arrival depends both on the probability that cells are available on the inputs and the probability that a switch can accept the cells. We first find the probability of accepting a cell i.e that a switch gives a grant. We assume local grant flow control, so this probability depends only on the occupancy of the buffer. We then use this to compute the probability that a cell actually arrives.

Let the flow control grants be specified by the grant vector $\mathbf{z} = [z_0, z_1, \ldots, z_d], z_i \in \{0, 1\}$ where the value 0 for z_m corresponds to no grant, and 1 corresponds to a grant for port m. Let $G(\mathbf{z}|\mathbf{s})$ be the probability that a switch gives a grant to inputs according to the grant vector $\mathbf{z}$ when in state $\mathbf{s}$. In the case of the shared buffer switch element the grants are given to i random inputs when there are $i \le d$ buffer slots available in buffer. Thus we have:

$$G(\mathbf{z}|\mathbf{s}) = \begin{cases} \dfrac{1}{\binom{d}{|\mathbf{z}|}} & |\mathbf{z}| = \max(d, b - |\mathbf{s}|) \\ 0 & \text{otherwise} \end{cases}$$

Let α_m be the probability that a cell is available to enter on an input port $m \in \{0, 1, \ldots, d-1\}$ of a switch, and let β_m be the probability that an output port receives a grant. Furthermore, we assume a cell arriving at an input m is destined to output i with probability $r_{m,i}$.

We model the potential impact of arrivals on input *m*, when the switch is in state $\mathbf{s}$ with a generating function $A(m|\mathbf{s})$ on the variables x_i where the coefficient of x_i is the probability that cell arrivals at input *m* add a cell to output buffer *i* for $i \in \{0, 1, \ldots, d-1\}$. $A(m|\mathbf{s})$ is given by:

$$A(m|\mathbf{s}) = 1 - \alpha_m + \sum_{i=0}^{d-1} \alpha_m r_{m,i} x_i \qquad \forall \mathbf{s}$$

Next we combine the arrivals on the input ports and condition the arrivals on the grant vector by letting $C(\mathbf{k}|\mathbf{z}, \mathbf{s})$ be the coefficient of the term $x_0^{k_0} \cdots x_{d-1}^{k_{d-1}}$ in the product $\prod_{\forall m,\, z_m = 1} A(m|\mathbf{s})$ where $\mathbf{k} = [k_0, k_1, \ldots, k_{d-1}]$. $C(\cdot)$ can also be thought of as the convolution of the cell arrival distributions represented by $A(\cdot)$ for the different inputs. Then the arrival probability is just the sum over all the possible grant vectors of the predefined probabilities, and is given by:

$$p(\mathbf{k}|\mathbf{s}) = \sum_{\forall \mathbf{z}} G(\mathbf{z}|\mathbf{s}) C(\mathbf{k}|\mathbf{z}, \mathbf{s}) \quad . \tag{9}$$

To facilitate the evaluation of $q(\mathbf{h}|\mathbf{s})$ we define the conditional probability $q(\mathbf{h}|\mathbf{z}, \mathbf{s})$, which is the probability that the departure of cells causes a state transition from state $\mathbf{s}$ to state $\mathbf{s}$-$\mathbf{h}$ given grants from the outputs that correspond to grant vector $\mathbf{z}$, as:

$$q(\mathbf{h}|\mathbf{z}, \mathbf{s}) = \begin{cases} 1 & h_i = \min(s_i, z_i) \qquad i \in \{0, 1, \ldots, d-1\} \\ 0 & \text{otherwise} \end{cases}$$

Then the departure probability is the sum over all the possible grant vectors of the conditional departure probability weighted by the associated grant or no grant probability, and is given by:

$$q(\mathbf{h}|\mathbf{s}) = \sum_{\forall \mathbf{z}} \left[\left(\prod_{\forall m,\, z_m = 1} \beta_m \right) \left(\prod_{\forall m,\, z_m = 0} (1 - \beta_m) \right) q(\mathbf{h}|\mathbf{z}, \mathbf{s}) \right] \quad . \tag{10}$$

The *potential throughput* of the switch and the *grant probability* are needed for the network analysis and can be calculated from the steady state probability. The grant probability is the probability that a switch gives a grant to input *m*, signifying that the switch can accept a cell that arrives

on the input in a given cycle, and is given by:

$$g_m = \sum_{\forall \mathbf{s}} \sum_{\forall \mathbf{z},\, z_m = 1} G(\mathbf{z} \mid \mathbf{s}) \pi(\mathbf{s}) \quad . \tag{11}$$

The potential throughput is the probability that a cell is available to leave from output *m* of a switch in a given cycle. For the shared buffer switch the potential throughput is the sum of the steady state probabilities for which the switch has cells for output *m* and it is given by:

$$t_m = \sum_{\forall \mathbf{s}} \pi(\mathbf{s}) \qquad \text{where} \qquad \mathbf{s} = [s_0, s_1, \ldots, s_{d-1}] \wedge s_m > 0$$

The number of states in the vector model is of the order $O(B^d)$ and thus the computational complexity grows very fast with increasing switch and buffer size. Therefore, it can only be used for very small switch elements. To reduce the number of states we consider an approximation model that has lower computational complexity.

4.3.2. Active Output Model

To obtain a more computationally tractable model, we model a single switch element using *d* coupled Markov chains, one for each virtual output buffer. A state $[s_i, \sigma]$ in the model for virtual output buffer *i* means that virtual output buffer *i* has s_i cells while the switch element has σ cells. More formally a state $[s_i, \sigma]$ in the active output model includes a state $[s_0, s_1, \ldots, s_{d-1}]$ of the vector model when $\sigma = \sum_{j=0}^{d-1} s_j$. Figure 4.2 shows an example of a 3 port switch element and how the states of the virtual output buffer for output 0 correspond to the states of the underlying vector model. The states for outputs 1 and 2 correspond, in a similar way, to horizontal and diagonal groupings of states from the underlying model. Figure 3.1 summarizes the notation used in this section.

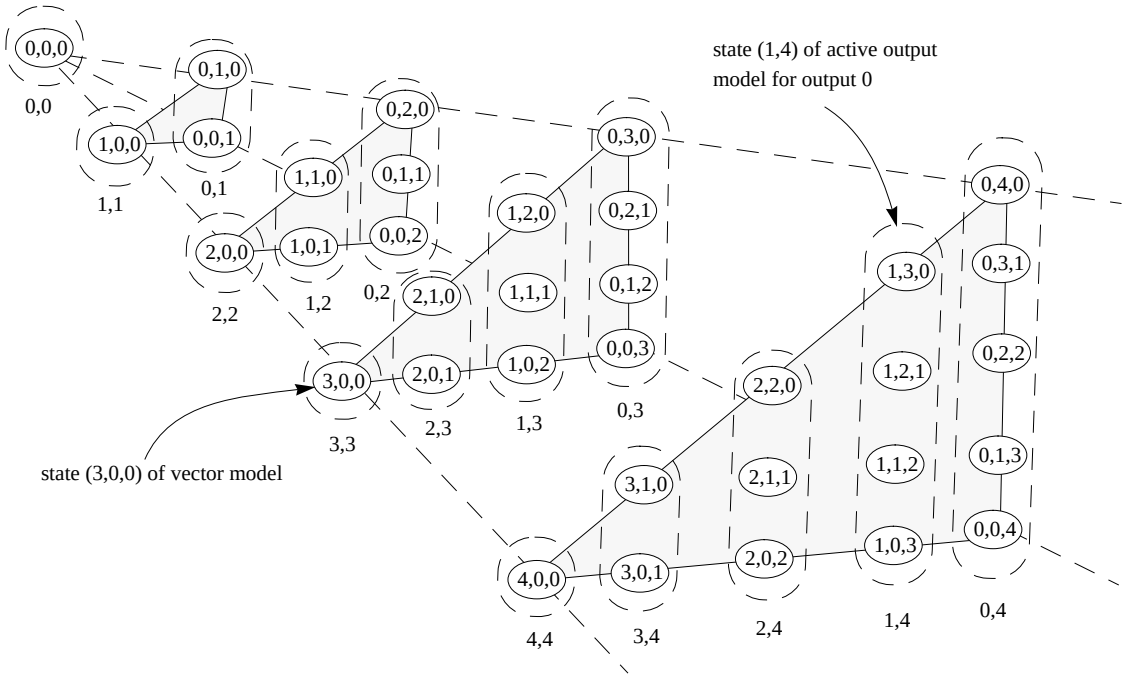


Figure 4.2: Example of mapping of states in the vector model to states in the active output model

Table 3.1: Summary of notation

Symbol	Comments
B	Number of shared buffer slots.
α_m	Probability that a cell is available to enter at an input *m.*
β_m	Probability that an output port receives a grant to transmit a cell at output *m.*
$r_{i,j}$	Probability that a cell arriving at input *i* is destined to output *j*.
$\pi_i(n, s)$	The marginal steady state probability that virtual output buffer *i* has *n* cells destined for it and there are *s* total number of cells in the shared buffer.
$\lambda_i(m, s_2 \mid n, s_1)$	Transition probability that Markov chain i changes states from (n, s_1) to (m, s_2)
$p_i(k_i, k \mid n, s)$	Arrival probability that cells arrive at the switch cause a transition of the switch state form state **`s`** to state **`k`**+**`s.`**
$q_i(h_i, h \mid n, s)$	Departure probability that cells departing the switch cause a transition from state **`s`** to state **`s-h.`**

Table 3.1: Summary of notation

Symbol	Comments
$G(z\|n, s)$	Probability that s switch gives a grant to z inputs when in state (n, s).
$A_i(m\|n, s)$	Generating function on the variables x_i, x and y representing the probability that there is no cell arrival, virtual output buffer i receives a cell, cell arrive but not for virtual output buffer i, and if grant is asserted or not.
$C_i(k_i, k\|z, n, s)$	Probability that k_i cells arrive for virtual output buffer i and total of k cells arrive, given that the state is (n, s).
$V_i(h_i, h\|n, s)$	Probability that h_i cells leave from virtual output buffer i and total of h cells departs from the shared buffer, given that the virtual output buffer is has a cell and there are cells in the shared buffer destined for z ports other than i.
$U_i(m\|n, s)$	Generating function on the variables x_i, x, y_i, and y representing the potential departure at output m, total departure, output m having a grant, and total number of grants.
$Y_i(z\|n, s)$	Probability that there are z output ports that have cells destined for them, given that the state is (n, s).
$L(m\|\dot{n}, s)$	Generating function on the variables x and y representing an active output and if it has a grant or not, given that the state is (n, s).
g_m	Probability that a switch gives a grant to input m.
t_m	Probability that a cell is available to leave from output m of a switch.

The marginal state of each virtual output buffer is represented with a tuple (n, s), where n represents the number of cells destined for the particular output, and s represents the total number of cells in the shared buffer. Let $\pi_i(n, s)$ be the steady state probability for virtual output buffer i and $\lambda_i(m, s_2|n, s_1)$ be the corresponding transition probabilities. Then the steady state probabilities can be expressed in terms of the transition probabilities:

$$\pi_i(m, s_2) = \sum_{\forall s_1} \sum_{\forall n} \pi_i(n, s_1)\lambda_i(m, s_2|n, s_1) \quad . \tag{12}$$

Let the arrival probability $p_i(k_i, k|n, s)$ denote the probability that cells arriving cause a transition from state (n, s) to state ($n+k_i$, $s+k$) of the virtual output buffer i, assuming no departure Let the departure probability $q_i(h_i, h|n, s)$ denote the probability that cells departing cause a transition from state (n, s) to state ($n-h_i$, $s-h$) of the virtual output buffer i, assuming no arrivals. The transition probabilities can then be obtained as follows:

$$\lambda_i(m, s_2|n, s_1) = \sum_{\substack{\forall k, h \\ s_2 = s_1 + k - h}} \sum_{\substack{\forall k_i, h_i \\ m = n + k_i - h_i}} p_i(k_i, k|n, s_1) q_i(h_i, h|n, s_1) \tag{13}$$

Once again we need to derive $p_i(k_i, k|n, s)$ and $q_i(h_i, h|n, s)$, but first we define the necessary probabilities. Basically we need to find how many cells arrive at a switching element and how many cells depart at the same time we need to know how many cells arrive at a virtual output buffer i and if a cell departed on output port i.

$G(z|n, s)$ is the probability that a switch gives grants to z inputs when in state (s,n), and is given by:

$$G(z|n, s) = \begin{cases} \dfrac{1}{\binom{d}{z}} & z = \max(d, b - z) \\ 0 & \text{otherwise} \end{cases}$$

The number of grants z is now a scalar compared with a vector for the Vector model.

Let $A_i(m|n, s)$ be a generating function on variables x_i, x, and y, representing potential impact of arrivals at input m. The constant term is the probability that no cell arrives, the coefficient of x_i is the probability that virtual output buffer i receives a cell from input m given that the state is (n,s), the coefficient of x is the probability a cell arrives, but not to virtual output buffer i, given that the state is (n,s), and the coefficient of y represents the fact that input i has grant asserted or

not. The generating function is given by:

$$A_i(m|n, s) = 1 - \alpha_m + \alpha_m (1 - r_{m,i})\, x + \alpha_m r_{m,i} x_i + y$$

Let $C_i(k_i, k|z, n, s)$ be the coefficient of the term $x_i^{k_i} x^{k-k_i} y^{d-z}$ in the product $\prod_{m=0}^{d-1} A(m|(n, s))$, the coefficient represents the probability that k_i cells arrive for virtual output buffer i and total of k cells arrive, given that the state is (n,s) and z grants were asserted. Then the arrival probability is given by:

$$p_i(k_i, k|n, s) = \sum_{\forall z} G(z|s) C_i(k_i k|z, n, s) \quad . \tag{14}$$

Our model makes it straightforward to determine if there is a cell destined to output port i from in the current state, but does not directly allow us to compute the total number of cells that depart from the shared buffer since we do not know how many of the other virtual output buffers have cells available. Define $Y_i(z|n, s)$ to be the probability that there are z outputs that have cells destined for them given that the switch is in state (n, s). The departure probability from virtual output queue i is given by:

$$q_i(h_i, h|n, s) = \sum_{\forall z} Y_i(z|(n, s)) V_i(h_i, h|z_i, z) \qquad z_i = \begin{cases} 1 & n > 0 \\ 0 & n = 0 \end{cases} \quad . \tag{15}$$

where $V_i(h_i, h|z_i, z)$ is the probability that h_i cells leave from a virtual buffer i and a total of h cells leave from the shared buffer, given that the virtual buffer has z_i cells in the buffer, and there are cells in the shared buffer that are destined for z output ports other than i.

Let $V_i(h_i, h|z_i, z)$ be the coefficient of the term $x_i^{h_i} x^{h-h_i} y^{d-z-z_i} y_i^{1-z_i}$ in the product $\prod_{m=0}^{d-1} U_i(m|n, s)$ where $U_i(m|n, s)$ is the generating function on the variables x_i, x, y_i, and y representing the potential departure at output m, total departure, output m having a grant, and total

number of grants available respectively. The generating function is given by:

$$U_i(m|n, s) = \begin{cases} 1 - \beta_m + \beta_m x_m + y_m & i = m \\ 1 - \beta_m + \beta_m x + y & i \neq m \end{cases}$$

To compute the number of virtual output buffers that have cells available we use all the marginal probabilities and assume that from the *s* total number of cells, the number of cells in each of the virtual output buffers is independent. Let $Y_i(z|n, s)$ be the probability that there are *z* output ports that have cells destined for them, it is represented by the coefficient of the term $x^{s-n}y^z$ in the product $\prod_{m=0}^{d-1} L(m|n, s)$. $L(m|n, s)$ is the generating function on the variables *x* and *y* representing an active output i.e output *m* has at least one cell destined for it given that there are *n* cells destined for output *m* and there are total of *s* cells in the shared buffer, given by:

$$L(m|n, s) = \pi_m(0|s) + \sum_{i=1}^{s} \pi_m(i|s)x^i y$$

The grant probability is the probability that a switch gives a grant to input *m*, signifying that the switch can accept a cell that arrives at the input in a given cycle, and is given by:

$$g_m = \sum_{\forall s, n} \sum_{\forall z} \frac{z}{d} G(z|n, s)\pi_i(n, s). \quad (16)$$

The potential throughput is the probability that a cell is available to leave from output *m* of a switch in a given cycle and is the sum of the steady state probabilities for which the switch has cells for output *m*, and it is given by:

$$t_i = 1 - \sum_{\forall s} \pi_i(0, s)$$

The number of states in the active output model is of the order $O(dB^2)$ which yields much smaller growth in complexity with increasing switch and buffer size than the vector model. Figure 4.3 shows a comparison of the number of states for both the vector model and the active output model.

The number of states for the vector model can be found by using the relation:

$$S_v(d, B) = \sum_{i=0}^{B} S_v(d-1, B-i) \qquad S_v(1, B) = B + 1$$

and for the active output model the number of states is: $S_a(d, B) = ((B^2 + 3B)/2 + B)^d$. The graphs show that the number of states for the active output model is larger than for the vector model only when $d = 2$ and it is much smaller when the size of the switch element increases to 4 and 8. This decrease in the number of states of the active output model makes it possible to evaluate the performance of larger switch elements.

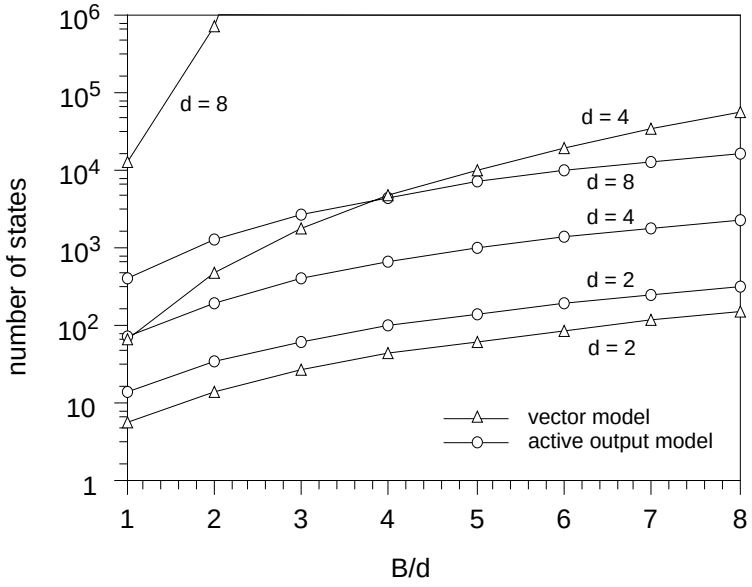


Figure 4.3: Comparison of the number of states

4.3.3. Comparison with simulation

In this section we compare results obtained with simulation to the active output model. We first study the effect of the normalized buffer size, B/d, on the maximum throughput. Our results are summarized in Figure 4.4 where the maximum throughput for several different switch elements sizes is plotted as a function of normalized buffer size. The curves show the analytical

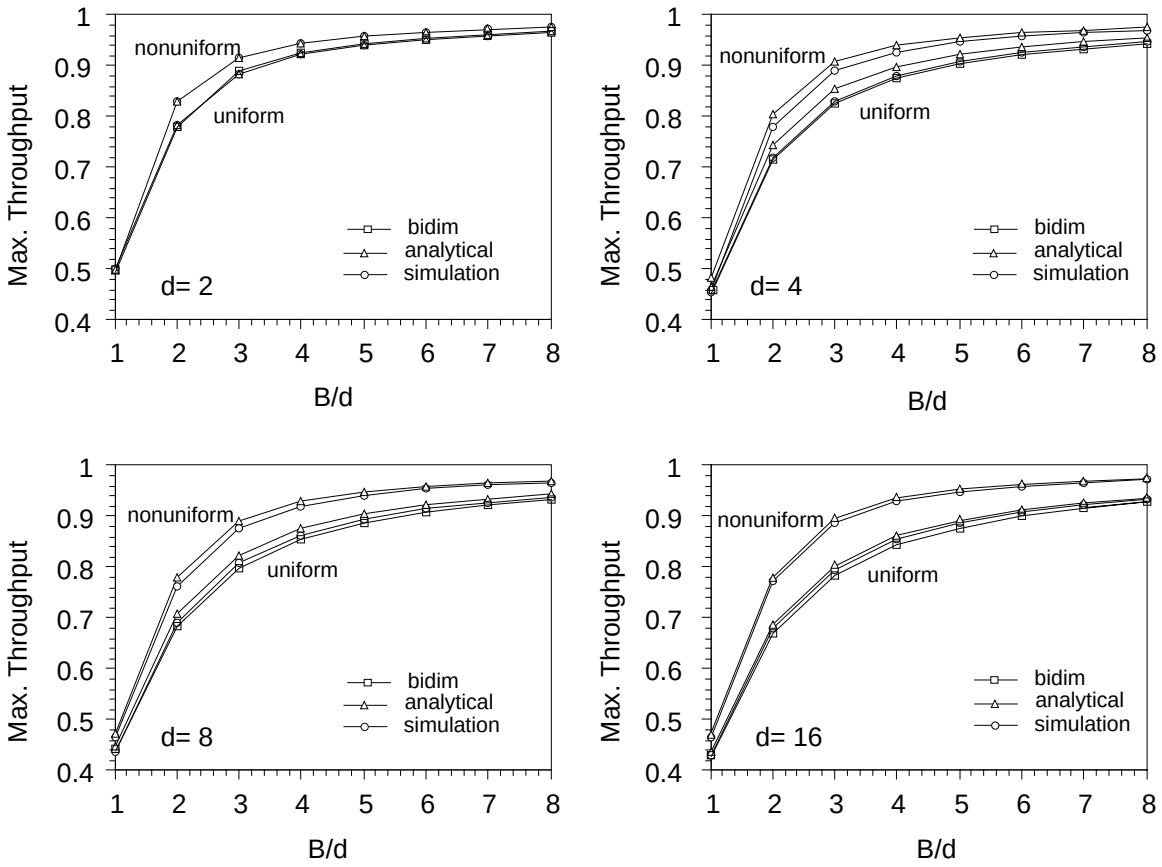


Figure 4.4: Throughput of single stage networks

results of the active output model for uniform traffic and non-uniform traffic with triangles, and corresponding simulation results with circles. In addition for uniform traffic we show the results for the bidimensional model presented in [8]. The nonuniform traffic matrix used in both the model and the simulation is of the following form:

$$\text{inputs} \begin{bmatrix} A_0 & 0 & \dots & 0 \\ 0 & A_1 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & A_{d-1} \end{bmatrix} \quad \text{outputs} \qquad \text{where} \qquad A_i = \begin{bmatrix} r_i & 1-r_i \\ 1-r_i & r_i \end{bmatrix}$$

and the r_i are chosen randomly from the interval [0,1].

The graphs show that the 2x2 switch element, (d = 2) model gives exact results but the larger switch elements overestimate the throughput slightly. In all cases the throughput increases as the

buffer size increases because of smaller cell loss. It can also be seen that the nonuniform load matrix gives a higher throughput than the uniform, this is due to fewer conflicts between cells destined to the same output port compared to the uniform case. The results for the analytical and simulator compare fairly well, but with the analytical model slightly over estimating the throughput. The difference between the simulation and the analytical model is smaller for larger buffer sizes and larger switch element size. This is because the virtual output buffers become more independent as they grow larger and as the number of inputs increases.

We have also compared the model and simulation with respect to cell loss probability. Figure 4.5 plots the cell loss probability for 2x2 switch elements and 8x8 switch element with

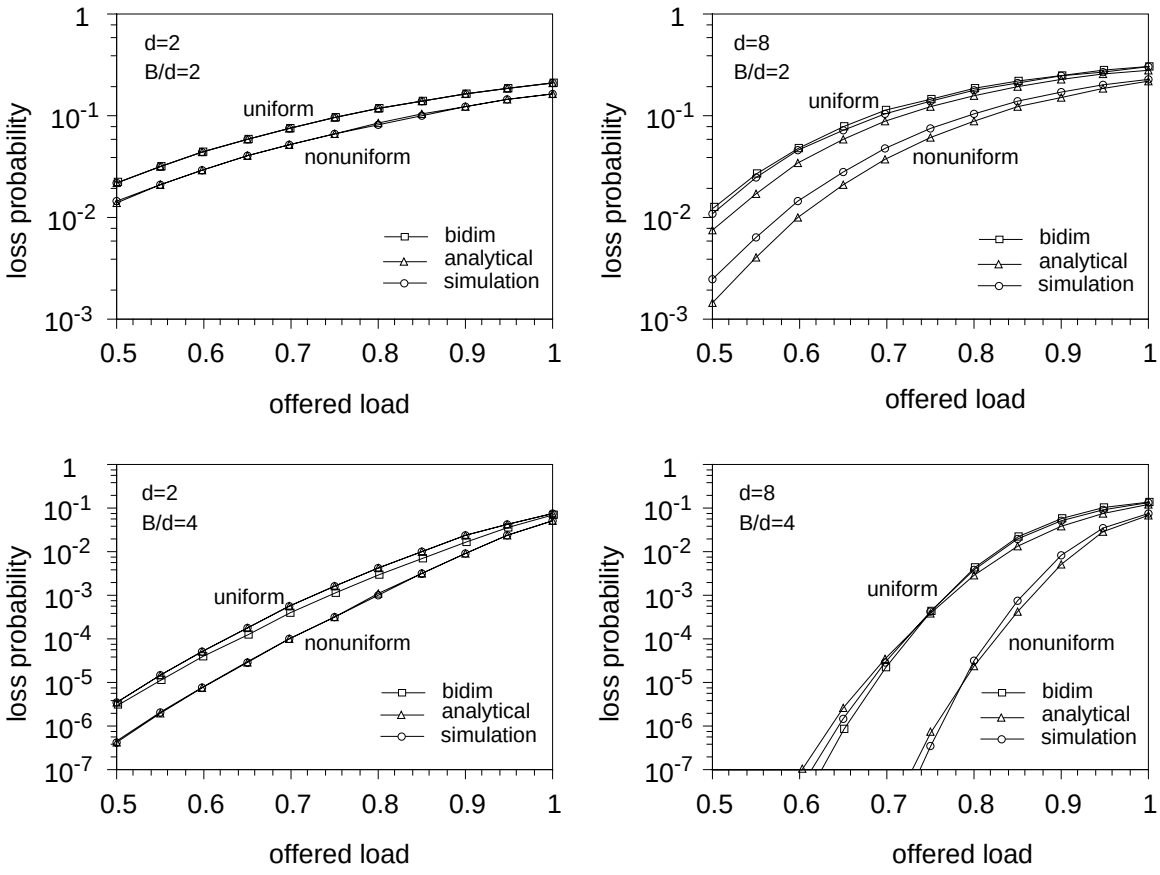


Figure 4.5: Cell loss in single stage networks

normalized buffer sizes of 2 and 4. The lines with the triangles plot the results from the active output model while the lines with the circles plot results obtained from simulation. The 2x2 switch element compares with the simulation as expected, since the active output model gives an exact result for the binary case. The 8x8 gives a slightly lower cell loss probability than the simulation indicates but the discrepancy decreases for the larger buffer size, as the buffers become nearly independent. Note that for single switch element the performance improves for non-uniform traffic due to the fact that cells are less likely to conflict by going to the same output port.

4.4. Analysis of Multistage Routing Networks

The explicit model of the state of a single switch element presented in the previous section can be used to analyze the queueing behavior of the entire delta network by assuming that the states of the various switches in a stage are independent. Let $\pi_{i,j}(\mathbf{s})$ be the steady state probability of virtual output buffer *j* of switch *i* and $t_{m,i}$ be the potential throughput for output port *m* of switch *i* and let $g_{m,i}$ be the acceptance probability for input port *m* of switch *i*. Let $\alpha_{m,i}$ be the probability of cell arrival to port *m* for switch *i*, and $\beta_{m,i}$ be the probability of receiving a grant for port *m* at switch *i*. If output *n* of switch *i* is connected to input *m* of switch *i* then $\alpha_{m,i} = t_{n,i}$, and similarly if output *m* of switch *i* is connected to input *n* of switch *k* then $\beta_{m,i} = g_{n,k}$. Obviously $\alpha_{m,i}$ and $\beta_{m,i}$ depend on the state probabilities of the neighboring switching elements. An iterative computational method is used in which first arbitrary initial values are assigned to the state probabilities, then values for $\alpha_{m,i}$ and $\beta_{m,i}$ are computed for all stages and finally these values are used to compute new balance equations for the Markov chain from which the new state probabilities are obtained. The iteration stops when convergence is reached.

Cells generated at each network input *i* are destined to network output *j* with probability $l_{i,j}$. The load matrix with elements $l_{i,j}$ describes the traffic pattern. We need to estimate the routing

probabilities for each switching element. This can easily be obtained from the global load matrix together with the interconnection pattern of the network. Figure 4.6 shows pseudocode for the evaluation of the routing probabilities of switch elements. In the pseudocode the variable *stages*

```
predicate FLOW() ;
    for  i ∈ [1, stages] ⇒
        for  j ∈ [0, rows – 1] ⇒
            for  in ∈ [0, d – 1] ⇒
                for  n ∈ [0, N – 1] ⇒
                    out = output(i, n)
                    if  i = 1  ⇒   localload (i, j, in, out, n) = globalload(j · d + in, n)
                       | i ≠ 1  ⇒
                            for  k ∈ [0, d – 1] ⇒
                                localload(i, j, in, out, n) = localload(i, j, in, out, n) +
                                    localload(i – 1, predrow(i, j, in), k, predout(i, j, in), n)
                            rof
                    fi
                    route(i, j, in, out) = route(i, j, in, out) + localload(i, j, in, out, n)
                rof
            rof
        rof
    rof
```

Figure 4.6: Procedure to calculate local load matrix

represents the number of stages in the network, *rows* corresponds to the number of switch elements in each stage, *d* is the switch element size and *N* is the number of inputs to the network. The function `output(i,n)` gives the port of the switch element in stage *i* from which the network output *n* is reachable and `predrow` and `predout` give the row and port of the preceding switch element respectively. All three functions depend on the network topology and are easy to evaluate. The routing parameters for a switch element in stage *i* and row *j* are then given by route(i, j, in, out). The global load matrix is globalload$(i, j) = l_{i,j}$ and the local load matrix

localload(i, j, in, out, n) corresponds to the load from input *in* to output *out* and destined to network output *n* at switch in stage *i* and row *j*.

Once the routing parameters for all the switch elements in the network are obtained, the iterative algorithm can be applied to obtain the performance of a multistage routing network.

4.5. Performance of Multistage Networks

In this section we look at the performance of the analytical model for multistage delta networks. The multistage network model using the active output model for each switch element can be applied to any kind of traffic pattern since it assumes an arbitrary general traffic pattern. We define several traffic patterns of interest for which the network performance is analyzed and compared with simulation.

4.5.1. Traffic patterns

Uniform traffic is defined as the traffic pattern in which every input port has the same rate of incoming cells and they are destined to every destination port with equal probability. Such a traffic pattern can be specified using the matrix shown in Figure 4.7. Any other traffic pattern is called nonuniform. We have selected two interesting traffic patterns to study the network performance from the many possible nonuniform traffic patterns.

One particular pattern is the single-source to single-destination traffic pattern in which each input source sends all its packets to a single output destination. Such a traffic pattern may cause extreme loads on common links shared by connections. A worst case performance of such a traffic pattern occurs for a delta network when cells on input port *i* are destined to output port *i*, corresponding to the identity traffic load matrix [59].

Another traffic pattern involves several destinations for cells at each input. A particular pattern that we chose to use assigns 4 random outputs to each input stream. Cells on an input are then

evenly likely to be destined to any of those 4 outputs as shown with the last matrix in Figure 4.7 The offered load is changed by varying the load intensity of each input. All the inputs have the same load intensity.

uniform

$$\begin{bmatrix} \frac{1}{N} & \frac{1}{N} & \cdots & \frac{1}{N} \\ \frac{1}{N} & \frac{1}{N} & \cdots & \frac{1}{N} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{1}{N} & \frac{1}{N} & \cdots & \frac{1}{N} \end{bmatrix}$$

identity

$$\begin{bmatrix} 1 & 0 & \ldots & 0 \\ 0 & 1 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & 1 \end{bmatrix}$$

random 8 x 8

$$\begin{bmatrix} 0 & 0 & \frac{1}{4} & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{4} \\ 0 & \frac{1}{4} & 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & 0 & 0 \\ \frac{1}{4} & 0 & 0 & \frac{1}{4} & \frac{1}{4} & 0 & \frac{1}{4} & 0 \\ \frac{1}{2} & 0 & \frac{1}{4} & 0 & 0 & 0 & \frac{1}{4} & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ 0 & 0 & \frac{1}{4} & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & 0 & \frac{1}{4} & 0 & 0 & 0 \end{bmatrix}$$

Figure 4.7: Global traffic matrices used in performance comparison

4.5.2. Performance comparison

Using the three traffic load matrices we examine the performance of several networks in terms of throughput, using the analytical model and simulations. Results obtained are summarized for comparison in Figure 4.8 where the throughput is shown as a function of offered load for 64 port delta networks. The networks are constructed from various switch elements of dimension d = 2, 4 and 8, and normalized buffer size, B/d = 3. The graphs plotted in the panels on the left compare the performance using the different traffic matrices for a particular network and the graphs on the right

compare the performance of the various networks for the same traffic matrix. The results obtained from the analytical model are plotted with triangles and corresponding simulation results are plotted with circles.

Overall the graphs show that when the uniform traffic matrix is used there is a discrepancy between the simulation and analytical results for high offered load. The difference is largest for small switching elements (d =2) and decreases when the switch elements increase in size (d = 8). This is due to the simplifying assumption of independence between switch elements which does not quite hold true because of the flow control between stages. For larger switch element size, the elements become less dependent resulting in performance predicted by the analysis. The uniform traffic gives the highest throughput of the traffic matrices as expected. The random and identity load matrices overload some internal links which results in lower performance. The throughput performance for the analysis using the random matrix is slightly higher than the simulation results and as before the difference is less for larger switching elements. The maximum throughput drops down to 0.6 for the random traffic matrix from 0.8 for the uniform traffic matrix. The use of the identity traffic matrix results in the lowest performance. This “worst case” performance can be compared with results obtained by Turner using a fluid flow approximation [59]. The worst case maximum throughput is 0.125 for d = 2, and 8, and 0.25 for d = 4, and agrees with Turner’s results for $D_{n,\,d}$.

$$\text{max.throughput} = \begin{cases} \dfrac{1}{\sqrt{n}} & \log_d(n) \text{ is even} \\ \dfrac{1}{\sqrt{\dfrac{n}{d}}} & \log_d(n) \text{ is odd} \end{cases}$$

where n is the number of inputs and d is the switch element size. Figure 4.9 (a) shows this better where the maximum throughput is plotted versus the network size for a delta network using switching elements with d = 2. The identity traffic matrix shows a stepwise function that

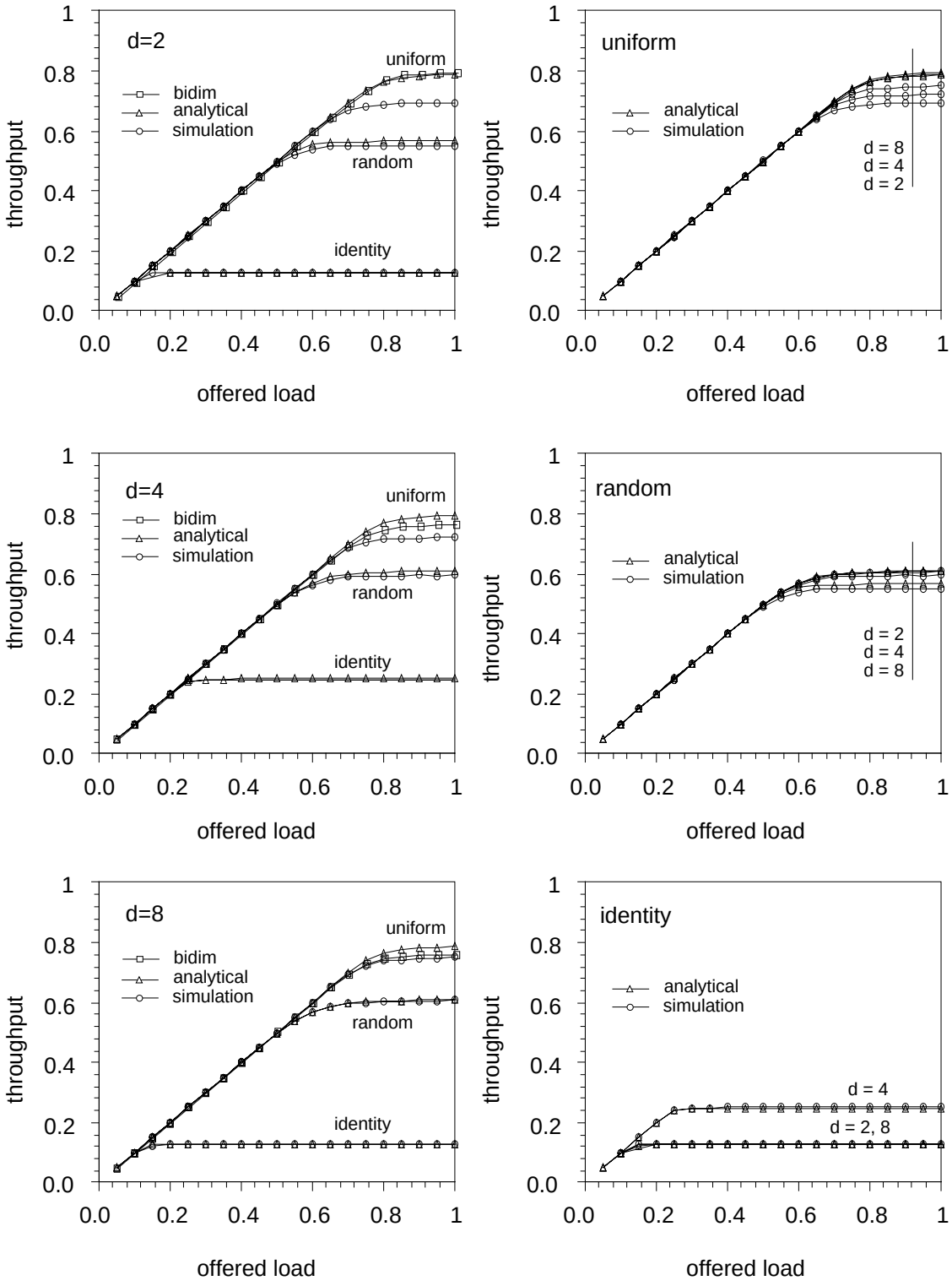


Figure 4.8: Comparison of analytical model with simulation

corresponds to the above equation obtained by Turner. If we look at the other traffic matrices we can see that over all the maximum throughput decreases as the network size increases. The uniform traffic matrix gives analytical results that diverge from the simulation as the network size and number of stages increase, because of interstage dependencies. Both the random load matrix and the identity matrix give analytical results close to the simulation results.

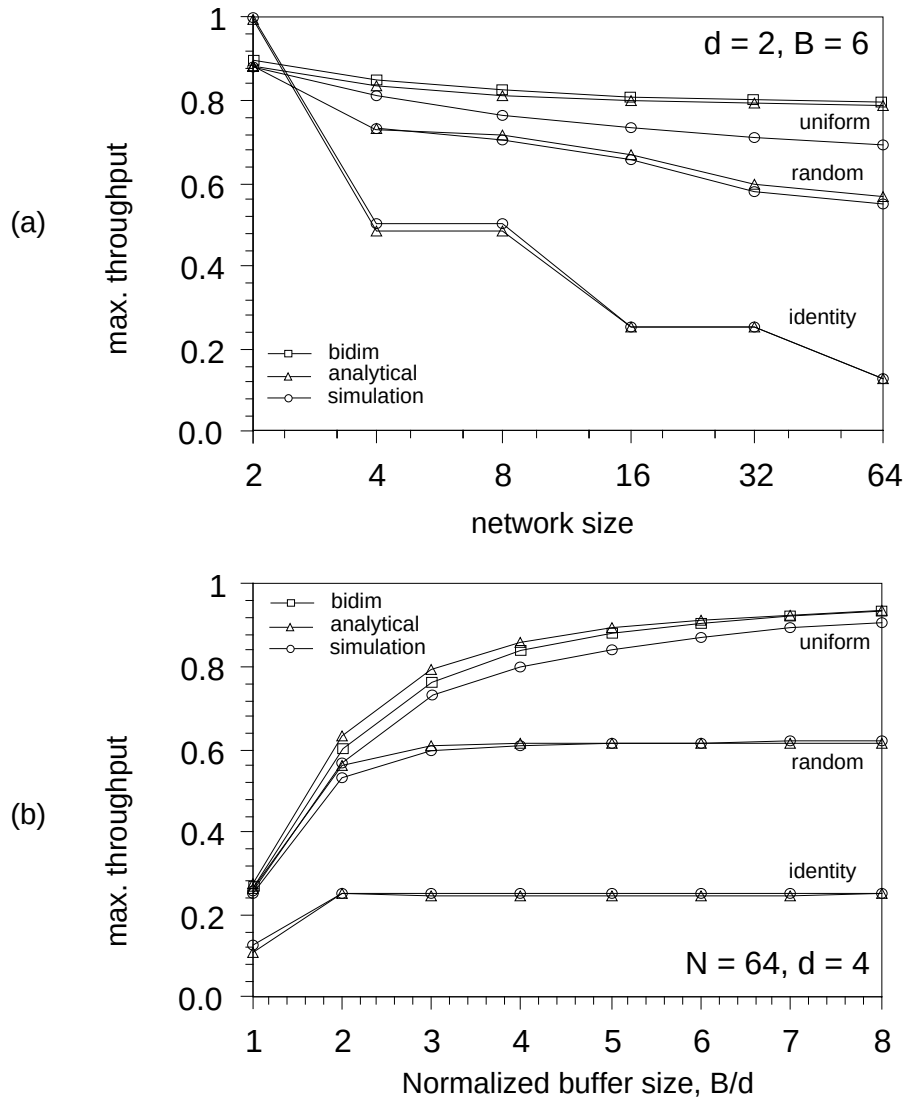


Figure 4.9: Effect of network and buffer size on throughput

Figure 4.9 (b) plots the maximum througput of a 64 input delta network with switch element size of d = 4 versus the normalized buffer size, B/d. The througput increases as the buffer size increases for all cases. The analytical results for the uniform traffic matrix get closer to the simulation as the buffer sizes increase. The random and identity traffic matrices give results that level off at a particular throughput. It is also noted that increasing the buffer size does not increase the maximum througput, because the traffic patterns result in a link overflow and the network can not handle any higher traffic load.

4.6. Remarks

In this chapter we have presented models to analyze the performance of shared buffered switching networks with non-uniform traffic patterns. We presented the vector model, which is an exact model for a single stage but is too computationally intensive to be used except for very small switch elements (d = 2). We then introduced an approximate model, the active output model, which reduces the complexity at the expense of accuracy. The single stage model in this case gives exact results for a 2 x 2 switch element and gives results close to the simulation results both for the maximum throughput and the probability of cell loss for larger switch elements. Finally we compared the performance of multistage networks for various traffic patterns and estimated the accuracy of the model by comparing the analytical results to simulation results. The results showed that the model gives a good approximation of the througput for nonuniform traffic and results similar to other proposed models for the uniform case.

5. SWITCHING NETWORK SIMULATOR

The design and analysis of switching systems require development and evaluation of extremely complex models. The different approaches that can be followed in the modeling process are either analytical analysis or simulation. The main advantage of the analytical approach is that closed form solution provide explicit relationships between performance measures and input parameters, which are very helpful in understanding the behavior of the systems. In practice however, closed form or numerical solutions can be derived only for simplified models.

Simulation can be used for the evaluation of these complex systems and can give accurate results for detailed models. Each simulation run is a random experiment that expresses some trajectory through the set of possible states. By repeating such experiments, or equivalently by running the simulation for a sufficiently long period of time, we can obtain accurate performance measures. We contend that general simulation packages are not suitable for simulating switching architectures. There are essentially two reasons for this. First, the construction of models of large switching systems using general purpose packages can be extremely tedious and time-consuming. Second, the execution speed of large simulation models is generally too slow for effective use. This led us to the development of a simulator specific for switching simulations.

We have used an object-oriented approach with the goal of using visual interactive simulation for network performance analysis. Visualization plays an important role in gaining a better understanding of switching systems by providing information on how systems actually operate. The visual approach provides tools that support the enhanced visualization of simulation objects such as static graphics for viewing the networks and animated graphics for viewing transaction movement, snapshots of network states, and evolution of statistics. A visual simulation environment has several advantages over the traditional programming approach [41]. The user deals with concrete

visual objects; conceptualization and model specification are closely woven, instead of being separated by a process of abstraction. The style of visual simulation provides a new dimension of understanding, unavailable in traditional simulation modeling. Extensive graphics can greatly aid in modeling specifications, while the interactive capability can cut down considerably the time and effort spent in debugging, testing, and interpreting the results, especially when different models are being compared.

Performance tools for switching systems are needed by many; our concern is to minimize the unnecessary overhead and duplication of work by having available a flexible tool, to which one can add functionality and extend rather than wasting time and effort starting from scratch. Switching system designers and researchers can use such a tool to explore different architectural alternatives much more easily than they could otherwise. Such a tool can also be helpful for people who would otherwise be reluctant to use simulations or performance analysis and as a teaching aid for students. Furthermore it can be used by network managers to help them decide how to configure a network, in support of particular traffic requirements.

The remainder of the chapter is organized as follows. In the next section we discuss some related work. Section 5.2 describes some of the design issues involved with a general switching system analyzer. The framework of the tool is presented in section 5.3. In section 5.4, we discuss the visualization aspects of the tool, and its implementation and command language in section 5.5 and 5.6.

5.1. Related Work

Simulation models are usually constructed in a modular fashion, defining blocks with well defined functions and clearly specified interfaces. This leads to structured model development and analysis techniques that can be assisted by software tools whose main components are the block

library, a simulation engine, and a language for the description of the model in terms of the block library. Simulation software for communication networks of this kind has been proposed and implemented. Examples include PET [7] and BONeS [18]. Both tools allow their users to construct simulation models of communication networks and conduct traffic related performance studies.

Most of the work devoted to the development of simulation tools for communication networks reported in the literature has been based on extended queueing networks. Among such existing tools are the AT&T Performance Analysis Workstation Q+ [25], and IBM's RESQ [50], both of which also have graphical interfaces. Low-cost, high powered workstations equipped with high-resolution displays have made it practical to employ extensive graphics to represent a model and observe its behavior through animation and dynamically evolving statistics. Several researchers have developed tools for communication networks aimed at the exploitation of such graphic capabilities, following the concept of visual simulation described in [41]. These tools includes several listed below all of which have graphical editors for system specification.

The Network Simulation Testbed (NEST) [23] is a graphical environment for simulation and rapid-prototyping of distributed networked systems and protocols. It provides a complete environment for modeling, execution and monitoring of distributed systems of various complexity.

GMA is a generic graphical modeling and analysis package for analyzing and predicting the performance of data communication networks [71]. It automatically provides a default parameters on the basis of model size and required performance measurement accuracy and has a user friendly interface to specific mathematical modeling routines.

INTREPID stands for an Integrated Network Tool for Routing, Evaluation of Performance and Interactive Design [15]. It is a framework of network design tools that can be used to design networks of variety of types, including circuit switched networks, packet switched networks and

hybrids. A collection of tools allows the user to explore design alternative by selecting the nodes and the links in the network the placement of network processor and the capacity of the links.

TOPNET, a tool for the visual simulation of communication networks [2], is a simulation software package exploiting the visual aspects of a simulation experiment. The communication network topology and architecture are described by drawings, the system dynamics are represented with a class of timed Petri nets, the simulation experiment is controlled through menus and buttons and results can be graphically displayed.

These tools have influenced our work especially with respect to exploiting visualization in a simulation environment.

5.2. Simulator Framework

The need for a simple general tool imposes several requirements. It is important that a performance tool be easy to use and simple enough so that it can be used by both a novice in performance analysis as well as by an expert. The switching network analyzer needs to be flexible enough to support performance evaluation of a variety of complex switching architectures.

The switching network analyzer is an object oriented simulation environment. Within the tool the description of the switching system is mostly given in graphic terms. This requires powerful graphics editing capabilities where network objects or components can be instantiated and manipulated in a graphical editor. The editor is the most important part of the tool and much of the functionalities of the tool are built around it. Other parts of the simulation environment are the network construction operators, the network component, simulation facilities, command language interface and finally the graph editor for displaying statistics and measurements.

5.2.1. Network editor

The main part of the simulation tool is the network layout editor, which can be used to display and directly manipulate *components*. The manipulation is done with so-called *commands* and *tools*. The layout editor and the tool palette are shown in Figure 5.1. The figure shows the interface with all the different parts labeled. Networks can then be constructed in the viewing area. The viewing area can be panned and zoomed to get a better view of the network. Furthermore one can have multiple viewing areas active showing separate parts of the networks.

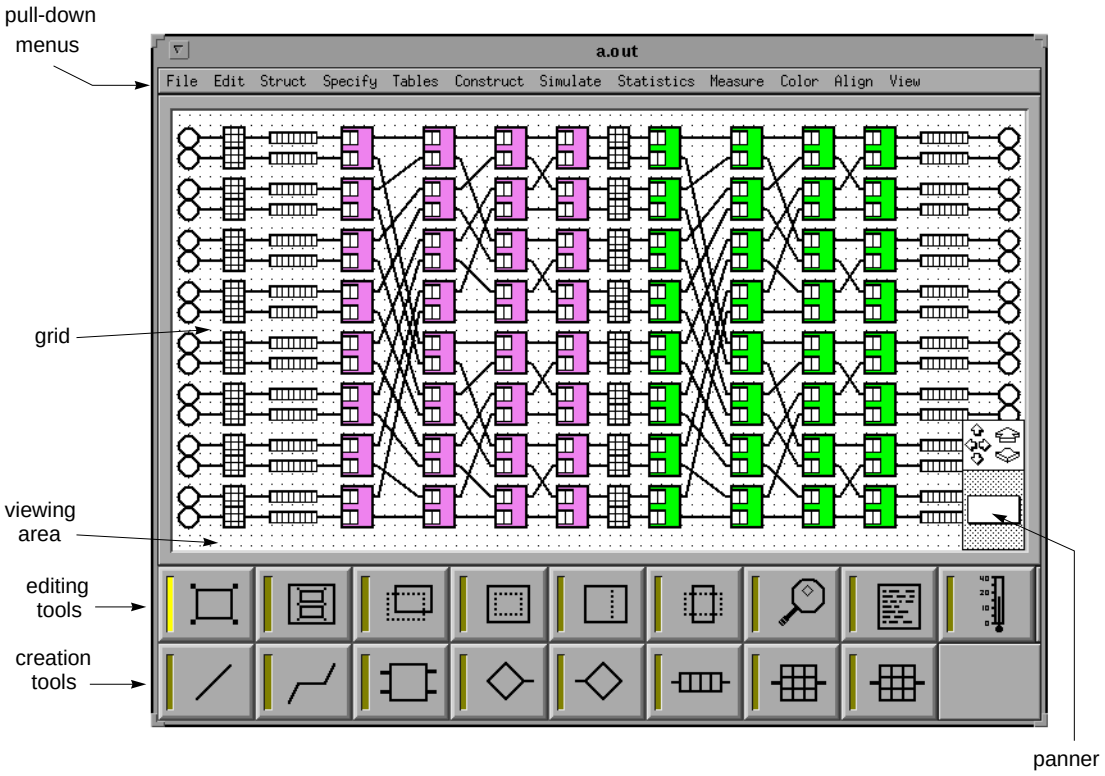


Figure 5.1: Network layout editor

On the bottom is a palette of creation tools that let us create instances of components. Tool selection controls the operation of the mouse pointer. The symbols shown in the tool window represent the basic network components: link, wire, switch element, source, sink, buffer, and lookup

table. New tools can be composed from existing components and dynamically added to the tool palette. To create a component instance, we simply engage its creation tool and then add an instance of that component in the layout editor with the mouse pointer. Using the components one can construct networks either manually or with the help of commands that are accessed through pulldown menus. The new networks can then be used as a basis for other networks by organizing them into a hierarchy of networks of increasing complexity. Besides the creation tools there are also edit tool in the tool palette that help edit network instances, either directly or in combination with pull-down menu commands. These tools are similar in concept to the creation tools, except they do not create new component instances. Rather, they affect existing components in the viewing area.

Above the viewing area are pull-down menus containing commands for editing different aspects of the network, for saving and retrieving networks that have been created, and for quitting the application. Besides editing commands, there are commands for network construction and commands for simulation of constructed networks. Finally, a row of indicators lies above the editing tool. These indicators display information about the editing session.

5.2.2. Components

To support simulation of a wide variety of switching architectures a rich set of components is required. The network designer is allowed to construct arbitrary networks from these components, and therefore the switching elements have to be able to operate together, as much as possible, independent of their internal structure or control mechanism.

Different combinations of flow control, buffering, and routing strategies need to be supported, and must work together in a seamless manner. This can be accomplished with a collection of different elements all conforming to the same external interface but with specific funtionality or a few

elements with general functionality that can be configured with different internal structure and control mechanisms. In our design, we chose the latter, primarily because the operation of a network can then be changed by just changing the properties of the network components, without having to construct the network again. Because it is impossible to account for all different switching systems, the tool must be designed in such a way so that it can be easily extended by adding new control mechanisms or new components and integrating them into the existing collection.

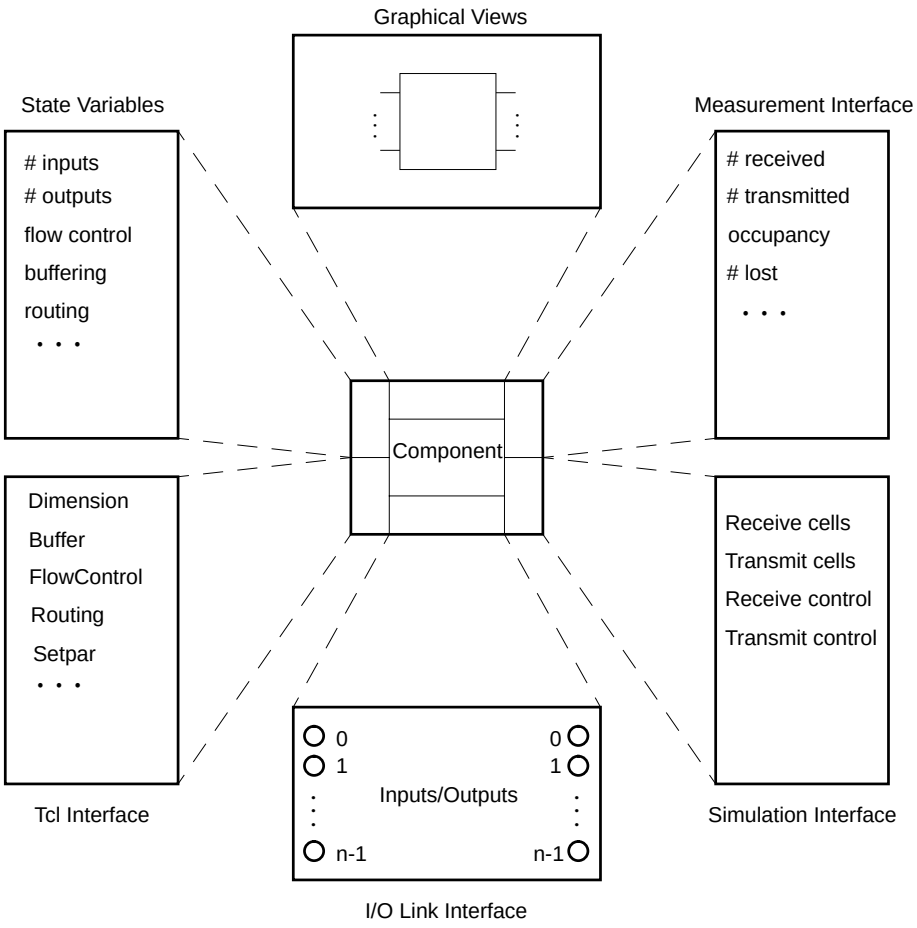


Figure 5.2: Component structure

Figure 5.2 shows the structure of a component. Each component specifies its own view to be displayed as a representation of the component. The view includes the input/output ports and possibly state information. The graphical view can then be manipulated in the network editor with the various editing commands, such as select, move, cut, paste, delete and duplicate. The state variables of the component are accessible through dialog boxes, where users can inspect and possibly modify the list of parameters pertaining to the internal structure and operations. After any changes to the state variables the component view and internal structure are updated. The input/output interface specifies the links connected to the component and are accessed during the network construction. During simulation, the component receives cells from the input links and transmits cells on the output links. Flow control signals are sent in the opposite direction from the data flow and are transmitted to the input links and received from the output links. The components have predefined actions which facilitate the flow of cells and control signals. These actions specify the internal operation of the component, how cells are processed within it and how they are routed and transmitted to the output links. Similarly, depending on the state of the component and the received signals, the flow control signals are asserted at the input links. A measurement interface is used to link the component model that counts events relevant for the computation of performance indexes to measures and statistics. Finally all the elements of the component are accessible through a Tool command language (Tcl) and can be specified through a command language script.

The main components are the switching element, source, lookup table, buffer and the sink component.

Switching element. The most essential and complex of the tool's components is the switching element. Switching elements are used to build multistage interconnection networks. The types of networks that can be constructed depends on the properties of the switching elements. The two main classes of networks are buffered and unbuffered networks. Buffered switch elements can use

input, output, shared buffering or some combination thereof. Buffered networks may or may not use flow control between switching elements in different stages. Flow control can be implemented by granting permission to send a cell ahead of time or by acknowledging receipt of cells. For grants, a switch signals to its upstream neighbor that it has a buffer space available, whereas in the case of acknowledgment flow control, the upstream neighbor sends the cell and the switch sends an acknowledgment back to the upstream neighbor as a notification of whether it could accommodate the cell or not. Furthermore, there can be several routing strategies since systems can route cells either by a fixed route or on a per cell basis, and in multipoint systems we need to be able to route or copy cells to several outputs.

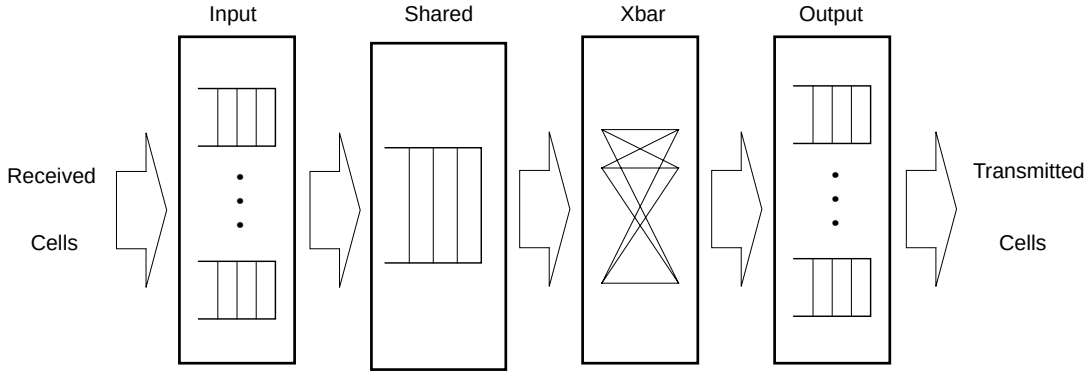


Figure 5.3: Internal structure of the switch element

The switching element model is very general and easily accommodates the variations among different elements. Figure 5.3 shows the internal structure of the switching element. The structure is divided in four sections, input, shared, xbar, and output sections. The input section contains input buffers, one for each input, the shared section contains a shared buffer, the xbar section includes the routing operation and the output section has output buffers, for each output. The switch element dimension, the buffering strategy and the buffer sizes, the routing and flow control

strategy is specified by the user through the state variables. Depending on the buffering strategy selected the three buffer sections can be included or omitted in the internal structure of the switch element or they are omitted. The received cells are sent to the first section present and if a section is missing the cells just go to the next section. All the buffering sections can operate as fifo buffers or with bypass queueing, furthermore the input section can use parallel queueing allowing all the cells in the input buffer to be considered for transmission. The flow control asserted to the upstream neighbor can easily be computed and depends only on the leftmost buffering section present. This structure allows us to model the different combination of buffering and unbuffered switch elements.

The xbar section is always included in the internal structure and performs the cell routing operation, that is, it selects the switch element output port on which the cell is to be transmitted. To add a new operation to the switch element all that is needed is to specify the routing algorithm for the new switch element; all the internal structure and cell flow between sections remains the same and will operate transparently with the different buffering and control options.

Source component. The source component models external input links. The links can operate at a lower speed than the internal switching fabric. Sources are modeled as bursty on/off cell generators, that feed into a buffer. Cells are generated and released from the buffer corresponding to the link rate. Each burst gets assigned a VCI according to a specified distribution, $P(\text{VCI} = i)$. If the switching system cannot receive the transmitted cell, the cell will be lost. Parameters for all the different generators can be set individually through the state variables. Figure 5.4 shows the basic model used for the source component. For a more general source model a real traffic trace can be used to generate the cells in the source.

Buffer component. The buffer component stores cells in a queue and is shown in Figure 5.5 (a). The total number of cells that can be to stored in the buffer depends on its size. The buffer can

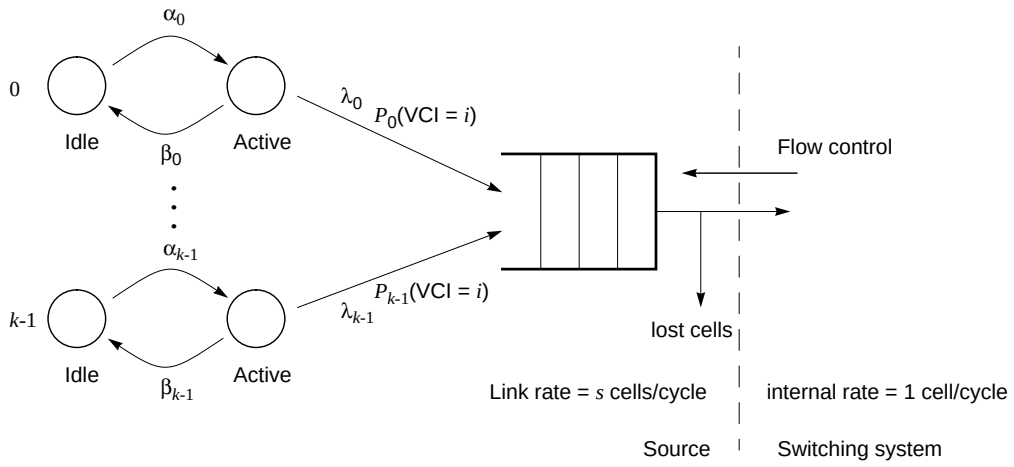
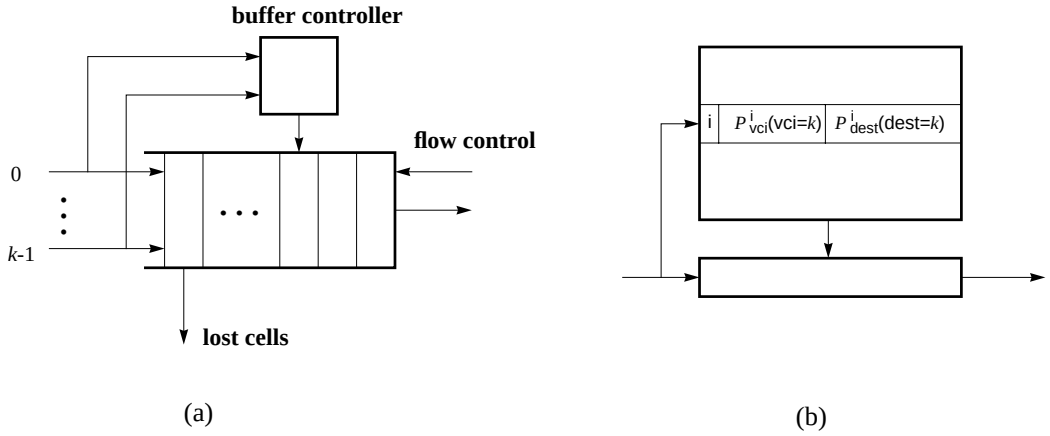


Figure 5.4: Source model

have several inputs corresponding to the number of cells it can accept in one cycle. The arriving cells are put in a particular order in the buffer depending on the service discipline. The service discipline can be FIFO, LIFO, by priority, or by age using time stamps and some threshold age. With the last one we can accomplish resequencing of cells that get out of order. Cells that are older than the threshold are discarded. Flow control can be asserted to the upstream neighbor depending on the buffer occupancy. If the buffer is full the arriving cells are discarded or negatively acknowledged.

Lookup table. A lookup table component is needed for channel translation at the input, broadcast translation for multipoint connections and fast buffer reservation [62]. The lookup component does not store any cells but passes them through with some modification to the cell header. Several fields in the cell can be used to index the table and different lookup tables can be used to assign values to particular fields in the cells. The assignment takes on values depending on a distribution associated with the field in the table. Figure 5.5 (b) shows a example of a simple lookup table that assigns a output VCI and output switch port from values defined by two different distributions. The different indexing values can have different distributions associated with them.



(a) (b)

Figure 5.5: (a) Buffer and (b) Lookup components

Other components include a sink which models the external output links, and merge and selector components are used in for recirculation in certain switch architectures.

Figure 5.1 shows a 16 port BPN switching system in the layout editor. The system is built from all the basic components. The switching fabric shows first the copy network and then the routing network separated by the broadcast translation lookup tables.

5.2.3. Network construction

In order to keep the tool flexible, one has to be able to build arbitrary network topologies. Arbitrary topologies can be constructed graphically by manipulating switch elements and connecting them together. Furthermore by providing some general construction operations, different network topologies can be specified quickly and easily. Once a network construction has been completed or when a network has been modified, the network should be simulated without any software compilation.

With the components described above the tool has the ingredients necessary for the construction of a complex switching system. However, to accomplish a general network construction an

additional ingredient, a *permutation*, is needed. Permutations are mappings of integers in the range [0, *n*-1] onto the same range. Permutations can be used to specify an interconnection of two networks. More specifically, let *f*: [0, *n*-1] $\rightarrow$ [0, *n*-1] be a permutation, N_1 a network with *n* outputs and N_2 a network with *n* inputs, then *f* specifies the interconnection pattern that for all *i*, $0 \le i < n$ connects output *i* of network N_1 to input *f*(*i*) of network N_2. Permutations can be generated and stored in a table, that later can be edited and/or selected. Furthermore permutations can be inserted in front of a network by assigning a permutation to the network. By doing this we change the input numbering of the network so that input *i* of the network becomes input *f*(*i*).

A generalized form of the perfect shuffle permutations that frequently occurs in network constructions can be generated automatically by the tool. This is accomplished by dividing the input range of *n* into d_2 groups of d_1 integers and the output range into d_1 groups of d_2 integers. The integers in an input group map to a different output group and vice versa. We can specify multiple connections between specific pairs of groups. In particular, we can specify that two groups be connected by r_1 set of links, where each set consists of r_2 consecutive links from within each of the groups. The r_1 sets are spaced evenly across the two groups.

Thus, the permutation has five parameters *n*, d_1, d_2, r_1, and r_2. The parameters must satisfy $d_1 d_2 = r_1 r_2 n$, where *n* is the domain of the permutation. We also require that both d_1 and d_2 are divisible by (r_1 r_2). The following mapping describes the permutation:

$$id_1 + k\frac{d_1 r_2}{r_1} + jr_2 + l \;\Rightarrow\; jd_2 + k\frac{d_2 r_2}{r_1} + ir_2 + l$$

where $0 \le i < d_2 / (r_1 r_2)$ $0 \le j < d_1 / (r_1 r_2)$, $0 \le k < r_1$, $0 \le l < r_2$.

Having described the general permutation we now turn our discussion to network construction. Network construction can be done by hand, by directly creating components using the component tools and then manipulating them with the editing tools. Alternatively, large networks can

be built more quickly with the help of several operators. The operators have predefined actions that help with the construction of large structured networks. The most important construction operations are concatenation and stacking, which are then used to do series, and parallel construction of networks.

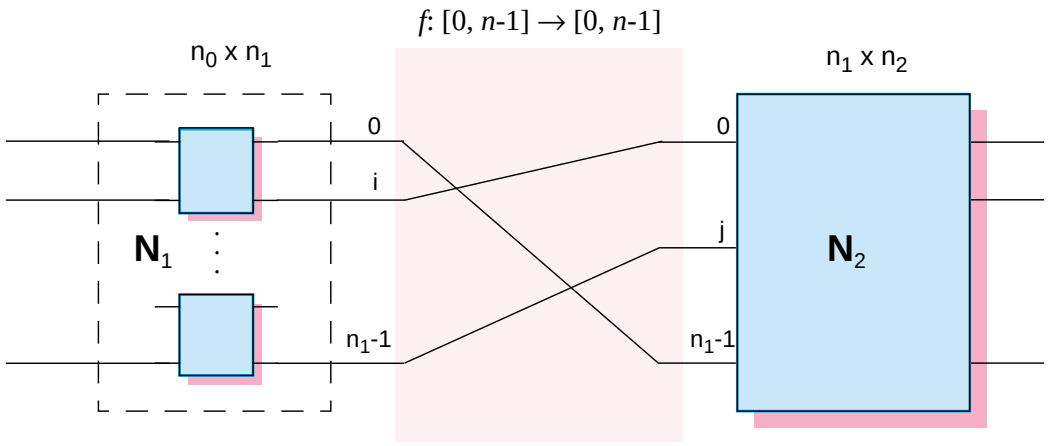


Figure 5.6: Stacking and concatenation of networks.

The *stack* operation makes several copies of a network and stacks them up so that the resulting network consists of a column of subnetworks. The *concatenate* operation connects two networks N_1 and N_2 together with a specified permutation. The number of outputs of N_1 must match the number of inputs of N_2. The permutation defining the interconnection pattern can be specified, thereby allowing arbitrary interconnection patterns. Figure 5.6 shows the concatenation of two networks where output i of network N_1 is connected via permutation f to input $f(i)$ of network N_2, network N_1 consist of a stack of subnetworks.

The series and parallel construction operators have been generalized to allow the construction of networks in which subnetworks are connected by multiple links instead of single links. Links can either be spaced as far apart as possible to maximize diversity of paths through the network, or by using multiple links in parallel for dilated networks. Using the stack and concatenate operator

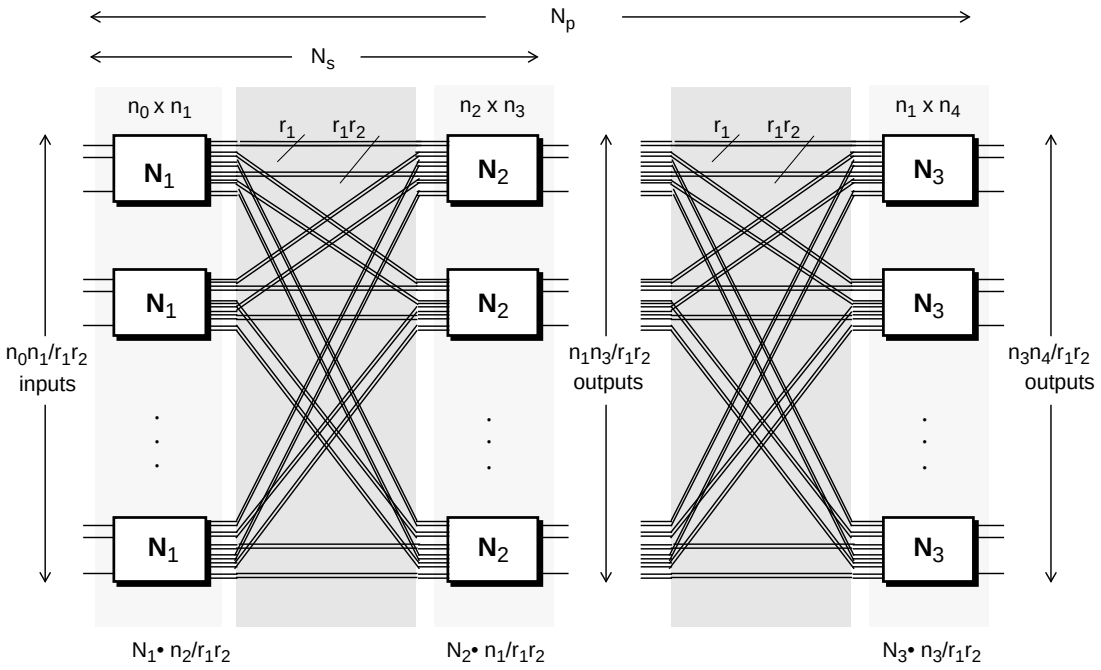


Figure 5.7: Series and parallel connection of networks.

the *series* command is implemented yielding the network constructions shown in Figure 5.6 for networks N_1 and N_2. The series command uses the permutation described above that connects r_1r_2 outputs of a particular copy of N_1 each to inputs of a different copy of N_2; this generates n_2/r_1r_2 copies of N_1 if there are n_2 inputs in N_2, and as n_1/r_1r_2 copies of N_2 if there are n_1 outputs in N_1. If N_1 is a (n_0,n_1)-network and N_2 is a (n_2,n_3)-network the resulting network N_s will be an $(n_0n_2/r_1r_2, n_1n_3/r_1r_2)$-network. The *parallel* command creates the network construction shown in Figure 5.6 from networks N_1 and N_2. It can be thought of as two independent series connections: first with the connection of N_1 and N_2 and then repeating again with connection of N_2 and N_3 using possibly a different permutation. Thus the parallel command generates n_3/r_1r_2 copies of N_1 if there are n_2 inputs in N_2, n_2/r_1r_2 copies of N_3 if there are n_3 outputs in N_2, and n_1/r_1r_2 copies of N_2 if there are n_1 outputs in N_1 and inputs in N_3. If N_1 is a (n_0,n_1)-network, N_2 a (n_2,n_3)-network, and N_2 a (n_1,n_4)-network the resulting network, N_p will be an $(n_0n_2/r_1r_2, n_3n_4/r_1r_2)$-network. Figure 5.6 shows

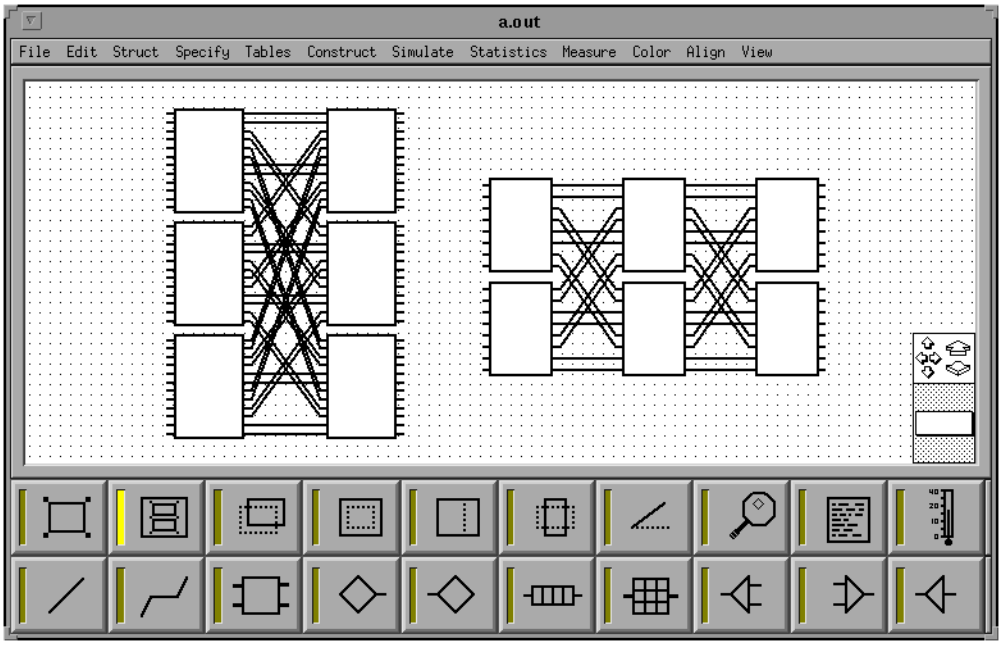


Figure 5.8: Examples of series and parallel connection.

examples of the series and parallel connection with $r_1 = r_2 = 2$ for switch element sizes of 12 and 8 respectively.

With these basic construction operators we can build any common network structure that has been proposed as a switching network by just selecting the right components and networks and then operating on them with the appropriate commands. The resulting network is then instantiated and displayed in the network editor. Several commands that construct specific topologies like delta, banyan omega and Benes networks are also available to help with the network construction. More complex construction commands can be easily implemented by the user by using the pre-defined operations through the command language.

Figure 5.9 shows a 16 input delta network used to construct the 16 port BPN switching systems in Figure 5.1. The network is constructed by using 3 series constructions each time with the resulting network and an additional switching element. The copy network and the routing network

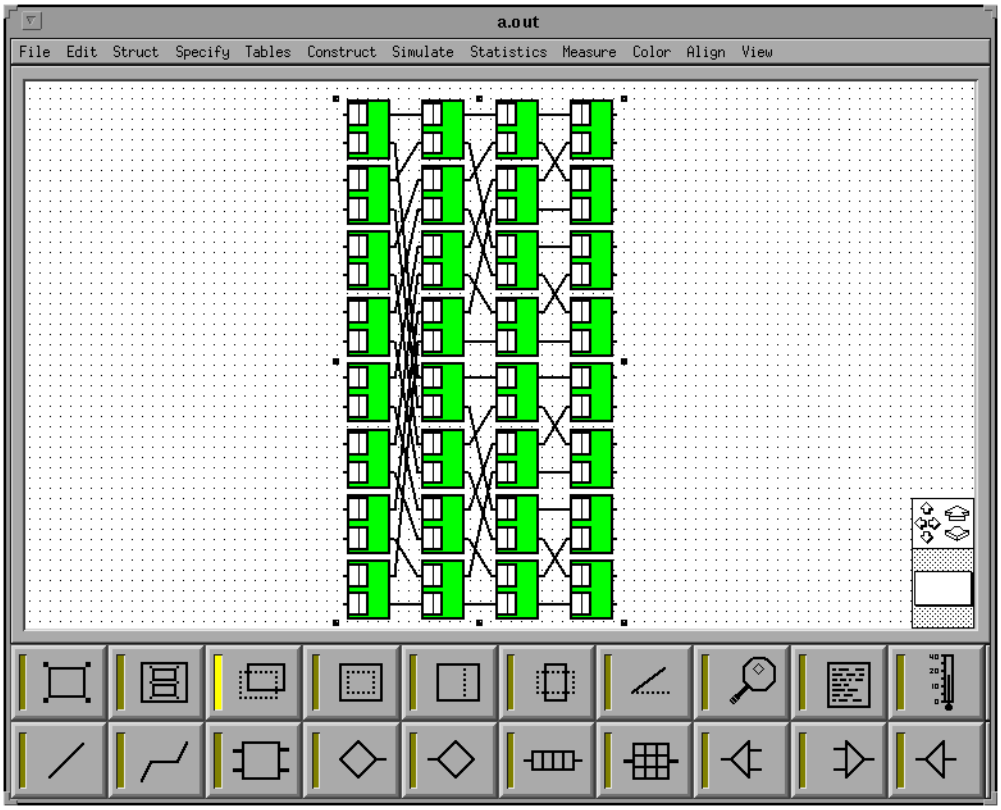


Figure 5.9: 16 input delta network constructed from 3 series constructions.

are constructed by just copying this network and assigning different properties to the switching elements. Stacks of sources, lookup tables, buffers and sinks are then concatenated all together resulting in the network in Figure 5.1.

5.2.4. Routing

The fact that the designer can construct arbitrary network topologies makes the routing of cells in the network from input ports to output ports difficult. The routing in the network can either follow a fixed path depending on a connection, or be on a per cell basis, using any path leading to the particular output. The switching elements must be able to independently select the right switch output leading to the destination network output, but to do this the switching elements need to

know about the interconnection pattern or at least the reachable set of network outputs from each of the switch elements. The simplest and most efficient method is to use routing algorithms that assume some predefined interconnection patterns. Such a routing algorithm can find the right output from the destination address of the cell but will restrict the network topologies that can be used. Most common networks have some regular interconnection pattern and the switch elements can independently route the cells to the desired network output. By using some function of the destination address the cell is sent to a particular switch output that leads to the right destination. The function that gives the *i*-th *d*-ary digit of the destination address is often used and is given by the following:

$$\text{output} = \left(\frac{\text{address}}{d^i} \right) \text{ modulo } d$$

where *i* corresponds to the column number of the switch element.

A more general way to accomplish the routing is to build routing tables at every switching element. The size of these tables depends on the size of the network and the switch element output size. For a large network it may be impractical if not impossible to include a large routing table with every switch element, because of memory limitations. Fortunately most of the networks of interest are fairly well structured, and have regular interconnection patterns and the same type of switching elements in a particular network stage. For these networks, the routing tables of each stage can be combined into one single table that can be shared by all the switch elements in that particular stage. Therefore it is possible to reduce the memory requirements drastically. However because an arbitrary network structure is possible we need to retain the possibility of assigning arbitrary routing tables to any switch element.

Many of the tables are of a certain form and can be generated automatically from several parameters; *n* number of network outputs, *d* number of switch element outputs, *g* number of outputs in a reachable output group from a switch element, *p*, the number of series output links, and *r*,

the number of diversity output links from a switch element. Figure 5.10 shows such a routing table were the shaded rectangle in column *i* and row *j* represents an existing route from the *i*-th switching element port in a given column to the *j*-th network output. As an example we use the 16-port delta network from Figure 5.6. Each stage would share a routing table, so the whole network would need four tables as shown on the right side of Figure 5.10. Note that the stages are numbered from right to left.

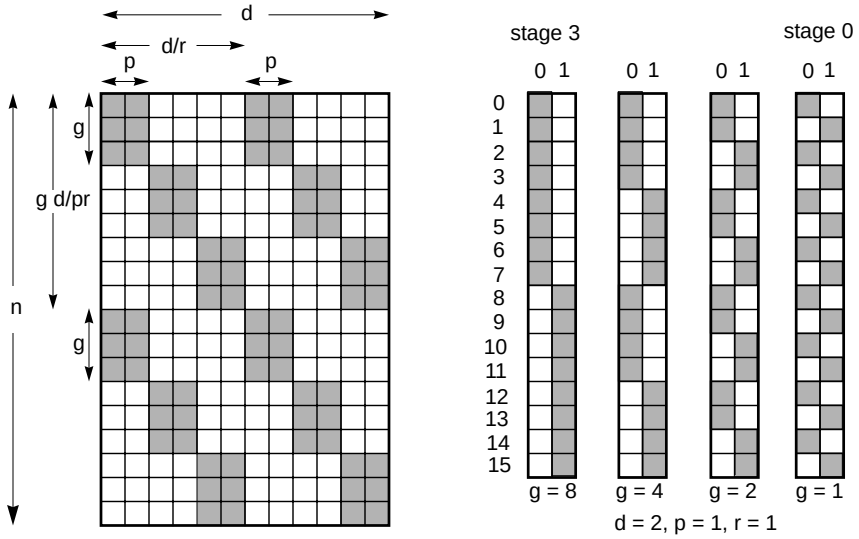


Figure 5.10: Automatic construction of routing table

By having both methods available the simulator can be efficient when network structure allows it but still provide general alternatives when unusual interconnection networks are being explored. The cost for the general alternatives will be in memory size and most likely in running time.

Similar considerations need to be made for copy networks in multipoint systems. Copy networks route or copy cells to several outputs. For a network with arbitrary construction a general copying algorithm may be impossible to find. The copying can be achieved by using knowledge of the network interconnection pattern or with the help of routing tables and by restricting the network topologies that can be used.

5.2.5. Network simulation

The method of simulation used is discrete time with activity scanning [24], which means that at every time step the list of components is searched to perform the actions required. This method is well suited to switching simulations since cells move synchronously from one switching element to the next and one cell time can be used as a time step. Furthermore, since there is flow of cells at most of the components at every time step the activity scanning approach is appropriate.

The actions in each component are divided into three phases. The first one makes sure that each component asserts the right flow control to its upstream neighbor. In the second phase, a component transmits cells at its outputs to the downstream neighbor. Finally in the last phase, each component accepts new cells at its inputs from its upstream neighbors. By dividing the action into three phases and controlling the order of the phases we can accomplish local and global flow control. The operations performed during these phases depend on the component state and its properties. Each component has actions defined during each of the three simulation phases. Modification of these actions allows change of the component operations. Once the switching systems have been constructed no software compilation is necessary before the simulation. Furthermore during simulation designers can change parameters and/or properties and continue the simulation. The user has control over the simulation. The simulation can be advanced one step at a time or over an arbitrary number of steps. The execution can be stopped, and then continued again, at any time.

5.2.6. Measurements and statistics

During the simulation, the state of the components is updated and some statistics are automatically collected. The data collection is limited by counting events in the links and the buffer occupancy, in order to ensure that data collection will not slow down the simulation. However enough data is collected to enable evaluation of the standard performance measures. Several classes of measurements are identified [2] according to their relation with the components:

External measurements are implemented outside components as simple probes that are inserted between components at the links in a completely transparent way. Only event counting and time series analysis can be obtained with measurements of this type.

Internal measurements require an interface to the components. The interface either records the appropriate internal events or makes queries about the events of interest. The component could have some fixed data collecting or be specified by the user when the network is built.

Tag-detag measurements require pairing of events at two different locations in the network. A typical case is the measurement of cell delays where cells are marked, time-stamped, and later recognized and the time stamp read.

Data collection can be divided up in periods giving us means to automatically evaluate confidence intervals for the statistics. Network components can be selected and their statistics printed out. The tool also implements a concept of *measures* which gives the user access to particular data internal to the components by querying the components. The measures can then be used for statistical and analytical purposes using visual representations by attaching them to graphs and plotting them against other measures. Furthermore one can combine several measures in a group and use the average of them or have functions operate on a collection of measures resulting in new measures that can then be used in the same way.

5.3. Visualization and interaction

Visualization is one of the major parts of the tool because it provides important feedback to the user during simulation. Through the means of such visual feedback the operation of the simulator can be verified. It also provides useful information for a better understanding of switching systems and for identifying many potential problems. Furthermore, this technique allows us to study transient traffic behavior such as short term congestion, in addition to the traditional steady state averages

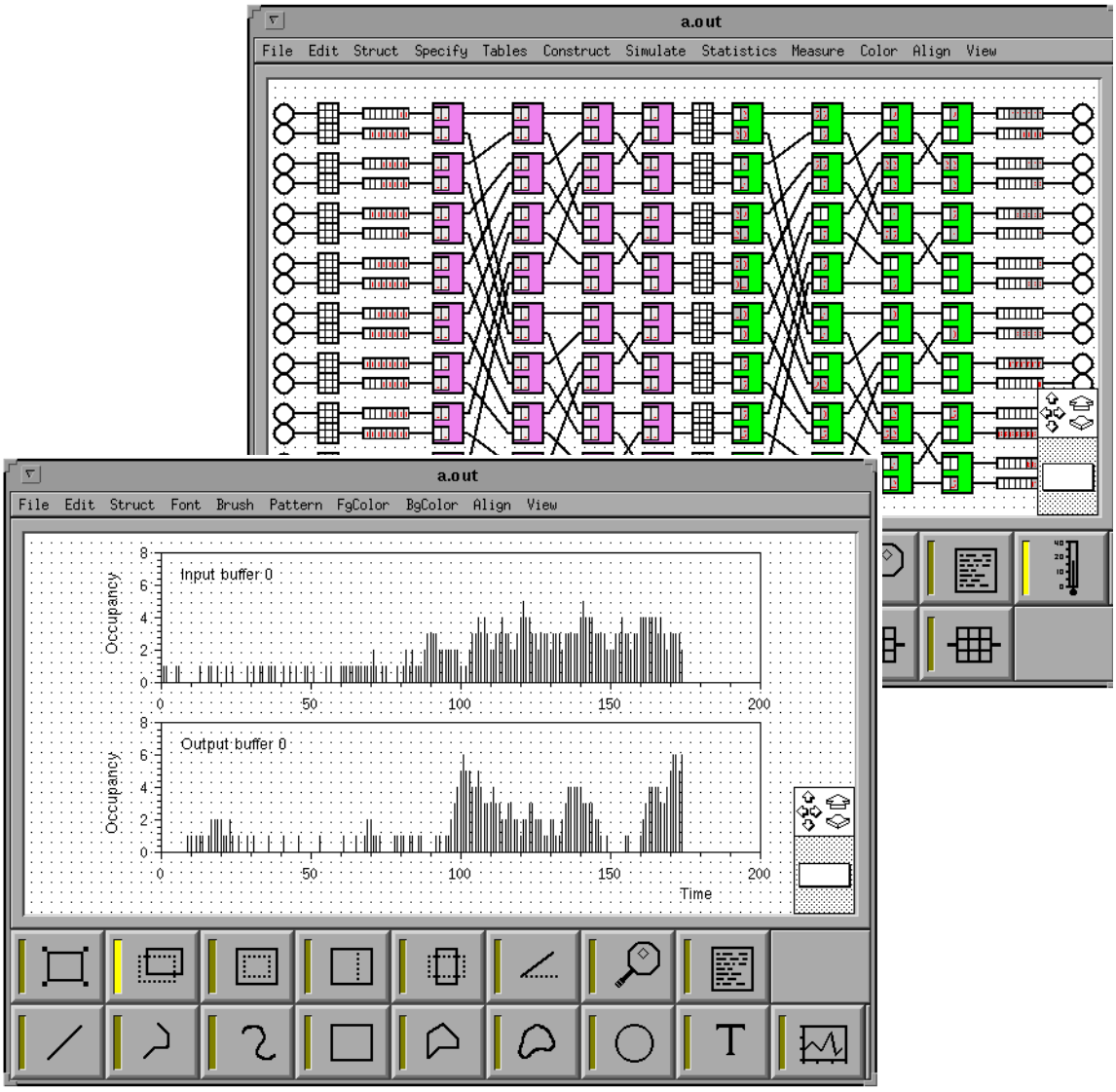


Figure 5.11: Network visualization

.Visual simulation appears to be a very valuable tool for gaining an understanding of how different traffic types and choices of design parameters can affect the simulation system. The highly interactive style of use that such a visual system allows is superior to conventional methods where the data is generated in bulk, then plotted and analyzed. We believe that by using this interactive type of simulation, the system designer can directly explore the effects of different parameters and concentrate more closely on choices of real interest.

Visualization is used in many areas of the simulation environment. Models are constructed in a graphical fashion. An abstract representation of the model is simply sketched as a drawing comprising icons that represent either network components or interconnection links. Users interact with data by manipulating their visual representation during processing. A sophisticated process of navigation is possible that allows users to dynamically modify or steer the simulation and computations as they occur. This allows the user to study the effect of changing parameters, or the representation of network components. Various state representations are possible. Users might be interested in instantaneous state representations for transient analysis or statistical distributions for steady state averages. The run time model animation depends on the level of detail of the system representation at which animation takes place. For large networks we have found that one cannot expect to display the detailed behavior of networks at one time. Summary information, and use of color and intensity information to indicate activity within the network have to be used. The use of multiple views also allow the simultaneous observation of portions of the system at several levels of detail. The presentation of results in graphic form may occur both at the end of the computation or simulation and also during run time by showing the evolution of performance measures as a function of the simulation time in specialized windows. Thus, as the simulation progresses, graphic representations of the current performance measures can be obtained. Statistics and other performance measures can be attached to the graphs and plotted in the graph editor as a function of simulation time or as histograms. Figure 5.11 shows some of the visualizations possible. The layout editor displays the network state by showing cells in the buffers and the graph editor we plot some performance measures of the network. In this example a instantaneous input buffer occupancy (top graph) and output buffer occupancy (bottom graph) are plotted as function of time.

By providing the user with the ability to control amount of visualization we can most efficiently select and represent the vast amount of information that is available in the simulation tool.

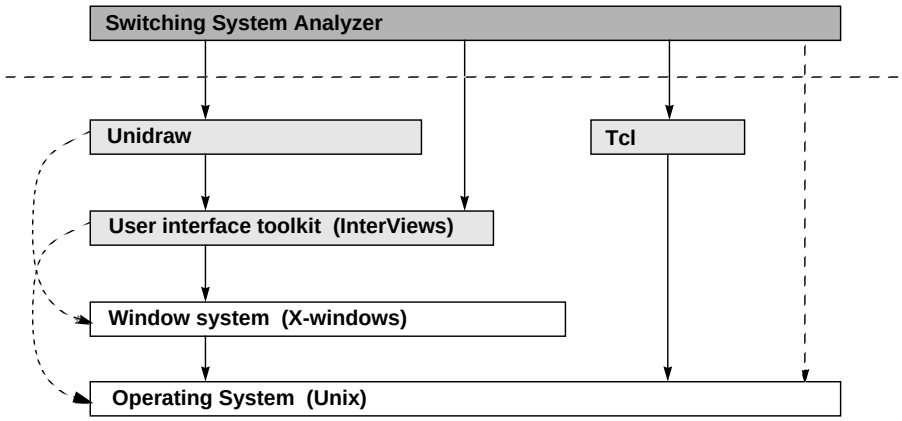


Figure 5.12: Layers of software underlying the performance tool

5.4. Implementation

The simulation tool is implemented using the C++ language, and runs on top of the X-window system. We are using the InterViews user interface toolkit from Stanford University. This toolkit was chosen mainly because it also uses C++ and because Unidraw [68], a graphical object editor library, is built on top of this toolkit. The Unidraw library is used for the basic graphical editing facilities and file manipulations of the performance tool. Figure 5.12 depicts the dependencies between the layers of software that underlie the performance tool. The operating and window system provide the lowest level support. Above the window system level are the abstractions furnished by the user interface toolkit including buttons, scroll bars, menus, and a framework for composing them into generic interfaces. Unidraw sits right below the performance tool contributing abstractions that are closely matched to the requirements of the graphical object editors and the connections between the components for the simulation.The simulation tool also extends a library of a command language, called Tool Command Language, (Tcl) and is described further in the next section.

The tool is based on an object oriented model in which objects encapsulate the attributes needed for abstraction. To accomplish this the tool relies heavily on classes, inheritance, and dynamic binding. By using the object-oriented approach, it is easy to extend the tool to include more components and additional functionality.

5.5. Command Language

The purely graphical modeling paradigm is very powerful. However it may become tedious to set component properties and to construct large switching systems. Therefore having an interpreted programming interface to the simulation tool can be extremely useful. The components can be manipulated and complete switching systems constructed by simple programming. Finally, by using the interpreted language to specify the graphical interface of the simulator the user is able to extend the simulator by writing procedures to manipulate components and then add those to the graphical interface of the tool.

The simulator has an interface to a command language called Tcl. Tcl is an application-independent command language [44]. It is a C library package that can be used in many different programs. The Tcl library provides a parser for a simple but fully programmable command language. The library also implements a collection of built-in commands that provide general-purpose programming constructs such as variables, lists, expressions, conditionals, looping, and procedures. Individual application programs extend the basic Tcl language with application-specific commands. The Tcl library also provides a set of utility routines to simplify the implementation of tool-specific commands and has a simple and efficient interpreter. We have extended the Tcl commands to include an interface to InterViews and Unidraw. Furthermore all menu commands and editing tools can be accessed through Tcl scripts.

Since the graphical interface of the simulation tool itself is based on a Tcl script, it is relatively easy to modify or customize or extend the interface with new commands and improve the

functionality of the simulation tool. Secondly, Tcl provides a uniform framework for communication between tools, which makes it possible for different tools to work together.

Tcl scripts can be written for sequences of operations, additional commands and extensions. An example is an automatic construction of switching systems of different sizes and properties. Similarly it is possible to write scripts that do simulation batch runs for different parameters and traffic patterns. Figure 5.13 shows a sample Tcl script containing two such procedures. The first one will create a switching network ready to be simulated. The network is a delta network connected up with Bernoulli sources, lookup table and sinks. The network can be specified to be of arbitrary size made from different sized switch elements. The second procedure specifies how to simulate the network for different traffic load. Each load parameter in the load list is simulated for a number of warm-up steps and the statistics are cleared. The simulation is then run for a particular number of steps and user defined performance measures are printed out. New menus or buttons can be added to the user interface and when selected with the mouse they will execute the procedures. The user can also type commands on the command line resulting in execution of the script.

5.6. Simulation Performance

In this section we examine the performance of the simulation tool in terms of running time. We also compare the simulator with non-visual and less flexible simulator written specifically for a certain architecture in order to determine the overhead of the flexibility and generality built in the simulation tool. Since the visualization has a large speed penalty we compare the simulation tool without the visualization with a simulator written specifically for a Benes network with shared buffer switch elements. In the simulation tool we have two models for the switching elements. The first is a general switch element that has been configured with shared buffers, and the other is a switch element that only models shared buffer switches. Figure 5.14 shows the running time for a

```
proc Delta {n k d B} {
    set editor neted
    set edcomp [$editor getcomp]
    set comp1 [sourcecomp bernoullisource 56 606 6]
    set comp2 [vcilookupcomp lookup 85 600 97 612]
    $comp2 entry 0 fix0 fan0 $unif${n} LtGray
    for { set i 0 } { $i < $k } { incr i } {
        set route($i) [switchcomp ${i}stageswitch
            [expr 120+$i*100] 600 [expr 170+$i*100] 650]
        $edcomp append $route($i)
        $route($i) dimension $d $d
        $route($i) route RouteAddress
        set stage [expr $k-$i-1 ]
        $route($i) setpar $stage [power 2 $stage]
    }
    set comp4 [sinkcomp mysink [expr 194+($k-1)*100] 606 6]
    $edcomp append $comp1 $comp2 $comp4
    $edcomp notify
    ncopycmd setpar $n ""
    $editor setselection $comp1
    $editor execute ncopycmd
    $editor setselection $comp2
    $editor execute ncopycmd
    set net $route([expr $k-1])
    for { set i [expr $k-2] } { $i >= 0 } { incr i -1} {
        $editor setselection $route($i) $net
        set net [$editor execute seriescmd]
    }
    $editor setselection $comp4
    ncopycmd setpar $n ""
    $editor execute ncopycmd
    $editor execute slctallcmd
    $editor execute alignvctrcmd
    $editor execute concatenatecmd
}

proc RunBernoulli {loadlist warmup total editor} {
    foreach load $loadlist {
        set sourcelist [ info commands bernoullisource?* ]
        foreach comp $sourcelist {
           $comp capacity $load
        }
        runonlycmd setpar [expr $warmup/$load]
        $editor execute runonlycmd
        $editor execute clearallstatcmd
        runonlycmd setpar [ expr $total/$load]
        $editor execute runonlycmd
        $editor execute printmeasurecmd
    }
}
```

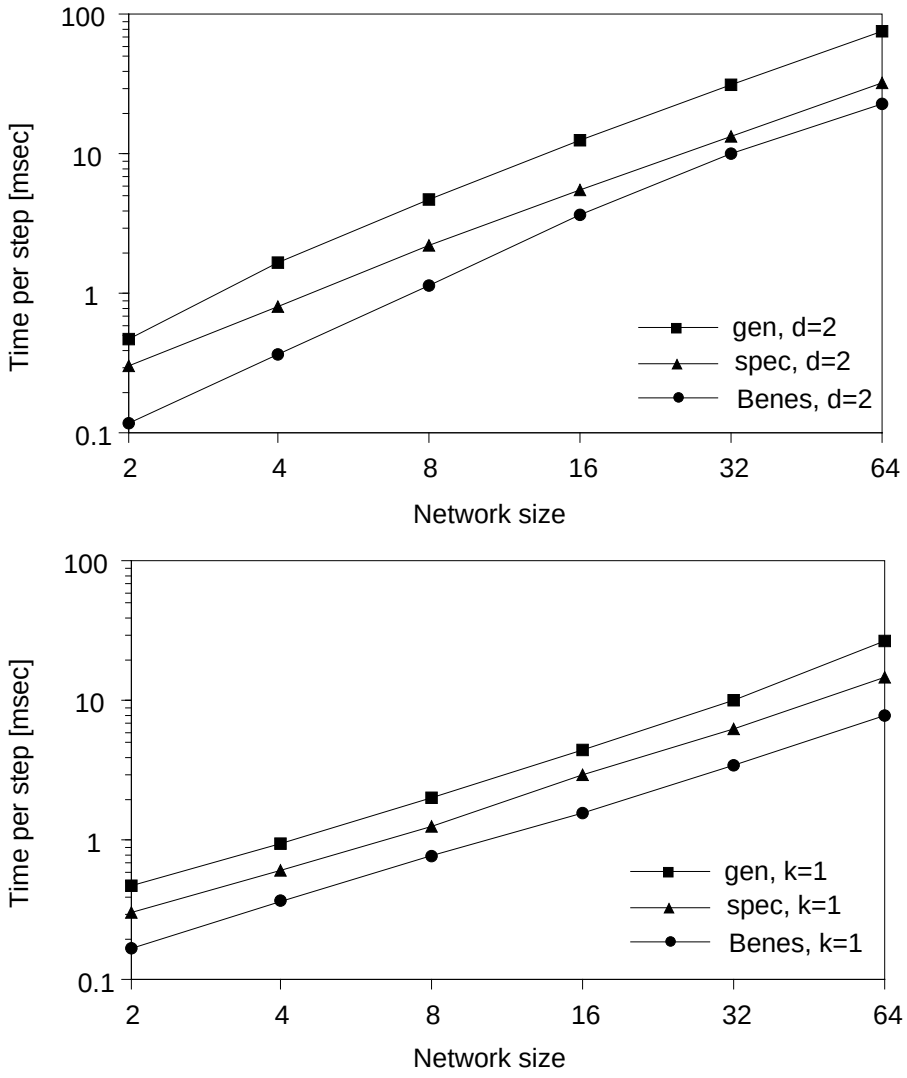


Figure 5.14: Simulation running time

single simulation step on a Sun Sparcstation 10. The graph on the top shows the simulation time for a binary Benes network with internal buffer of 8 slots, an input buffer of16 slots, and an output buffer of 32 slots. The running time increases fairly linearly as the network size increases. The general switch element (boxes) has the longest running time, about four times the Benes simulator (circles). Using the shared buffer switch element (triangles) improves the performance to less than two times of that of the Benes simulator. The difference decreases from a factor of two down to 1.2

as the network size increases. The graph on the bottom shows the running time for a single stage network with four times more buffer slots than the number of inputs. The graph shows similar behavior as before with little less running time for the same size network.

From the plots we can estimate that a simulation of 32 port network for 1 million time steps would take about 2.7 hours for the Benes simulator and 3.8 hours for the simulation tool Even though a factor of 2 is a significant increase in the running most people are willing to pay the price in running time when gaining such important features as ease of use and flexibility.

5.7. Remarks

We have presented a new general purpose tool for the simulation and evaluation of switching networks. This tool allows graphical manipulation of models, easy construction of network topologies, simulation without any additional compilation and visualization of the performance measures at run time. The tool is easy to use and can be used by a variety of people to construct and evaluate complex switching networks. The tool is flexible and efficient enough to support many different switching architectures and fast enough to provide a fundamental framework for the evaluation of such systems.

6. SWITCHING SYSTEM COMPARISION

In this chapter we evaluate the performance of several switching systems proposed for ATM. The study is aimed at evaluating and comparing the performance of some of the systems discussed in Chapter 2. All the systems are studied under exactly the same traffic conditions so their performance can be easily compared. The performance is obtained under three different traffic conditions: Bernoulli uniform traffic, Bernoulli non-uniform traffic, and bursty non-uniform traffic. We want to determine what increase in hardware complexity is needed for bursty traffic, and if besides larger output buffer sizes any additional complexity is required in order to obtain similar performance as for Bernoulli traffic.

Simulation is used to evaluate the performance of the switching systems via the switching simulation tool described in the previous chapter. This comparison study serves also to demonstrate the modeling capabilities, and flexibility of the switching simulation tool.

The organization of the chapter is as follows. We first discuss related work and then we describe the traffic model used. Next we present the simulation studies and the results. Each of the switching systems is first discussed separately and then we compare their complexity based on the performance requirements. The chapter ends with a summary.

6.1. Related Work

The architectures considered are among numerous that have been proposed in the literature [27][38][56][61][72]. Most of these proposals contain extensive performance data, but often comparison between architectures is hard because each proposal uses different traffic models with the exception of Bernoulli uniform traffic. The architectures can therefore only be compared in performance under such simplified traffic conditions. Comparisons between architectures tend therefore

to focus on qualitative, high-level design differences rather than concrete performance or cost differences [1][55]. A notable exception is the work by Zegura [70] comparing the cost of several ATM architectures on the basis of pin-limited chips for a particular performance. The performance was fixed to meet certain cell loss constraints under Bernoulli uniform traffic.

Pattavina [47] has studied the trade-off between ease of implementation and delay-throughput performance for non-blocking ATM switching architectures. The switches employed input, shared and/or output queueing. Hardware issues were discussed with respect to implementation feasibility and performance issues pointed out traffic bottlenecks of the different structures.

6.2. Traffic Models

We consider three types of traffic models for the simulation comparison. The first is Bernoulli arrivals of cells with output destinations uniformly distributed. Cell arrival at each input is independent and at the same rate characterized by the offered load ρ. The probability that a cell arrives in a given cell cycle is equal to the cell rate.

The second traffic model uses the same Bernoulli arrival process but the traffic pattern is non-uniform.This pattern involves several destinations for cells at each input. A particular pattern that we choose to use assigns four random outputs to each input stream. Cells on an input are then evenly likely to be destined to any of those four outputs. The cell arrival rate, ρ is the same for each input and the inputs are independent.

Finally we use a bursty arrival process. A source alternates between active and idle periods of geometrically distributed duration. Cells arrive with a rate λ at an input port during an active period and are considered to belong to the same burst. All cells from a bursty source are destined for the same output port. The duration of the active period is characterized by a parameter α. The probability that the active period lasts for a duration of *i* time slots is:

$$P(i) = \alpha (1 - \alpha)^{i-1} \qquad i \geq 1$$

The mean burst length is given by:

$$E_A[i] = \sum_{i=1}^{\infty} iP(i)$$

The idle period is geometrically distributed with parameter β. The probability that an idle period lasts for *j* time slots is:

$$Q(j) = \beta(1-\beta)^{j-1} \qquad j \geq 1$$

The mean idle period is given by:

$$E_I[j] = \sum_{j=1}^{\infty} jQ(j)$$

Given α and β, the offered load of a source, ρ can be found by:

$$\rho = \frac{E_A[i]}{E_A[i] + E_I[i]} = \frac{\alpha}{\alpha + \beta}$$

The total offered load at an input is then the sum of the load of all the sources at that input. Each input has a superposition of four bursty sources and the destination of each bursty source is uniformly chosen among four output ports corresponding to the non-uniform traffic used. An average burst length of 20 is considered for all the bursty sources, and the each source has maximum peak rate.

6.3. Switching System Performance

We consider five specific switching architectures: the Knockout, Tandem banyan, Sunshine, Lee's, and shared buffer Benes network. Most of these switching systems are discussed in Chapter 2, but we will review them briefly for completeness

In our simulation study we use 16 x 16 port switching systems with link rate of 155Mbs. The statistics of about 16 million cells are collected for each data point after a warm-up period. Thus, cell loss probabilities smaller than 10^{-7} are not measurable, and cell loss probabilities below 10^{-5} could contain non-negligible errors.

6.3.1. The Knockout switch

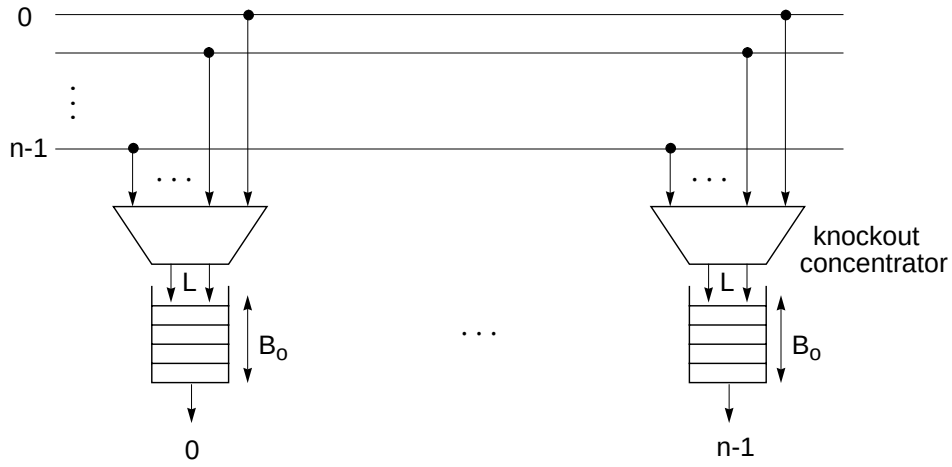


Figure 6.1: Knockout switch

The first switching system we consider is the Knockout switch [72], which can be considered as a single stage crossbar network. The structure of the Knockout switch is shown in Figure 6.1. The network consists of n^2 disjoint paths, connecting each input to an output. At each output the input cells are filtered and a knockout concentrator directs the cells that are destined to the output into the output buffer. The output buffers need to be able to accept up to L cells at every cycle. The performance of the knockout switch depends on the concentration ratio n:L and the output buffer size, B_o.

Figure 6.2 shows the performance of a 16 port Knockout switch, with L = 8, as a function of the number of output buffers, for the three traffic models. The figure shows two columns of graphs and three rows. The graphs in the left column plot cell loss probability as a function of output buffer size, and the ones in the right column plot cell delay as a function of output buffer size. The graphs in the first row plot results for uniform Bernoulli traffic, the ones in the middle row plot results for non-uniform Bernoulli traffic, and the graphs in the bottom row plot results for non-uniform bursty traffic. Each graph has a family of curves for different offered load, which is indicated by the number next to each curve.

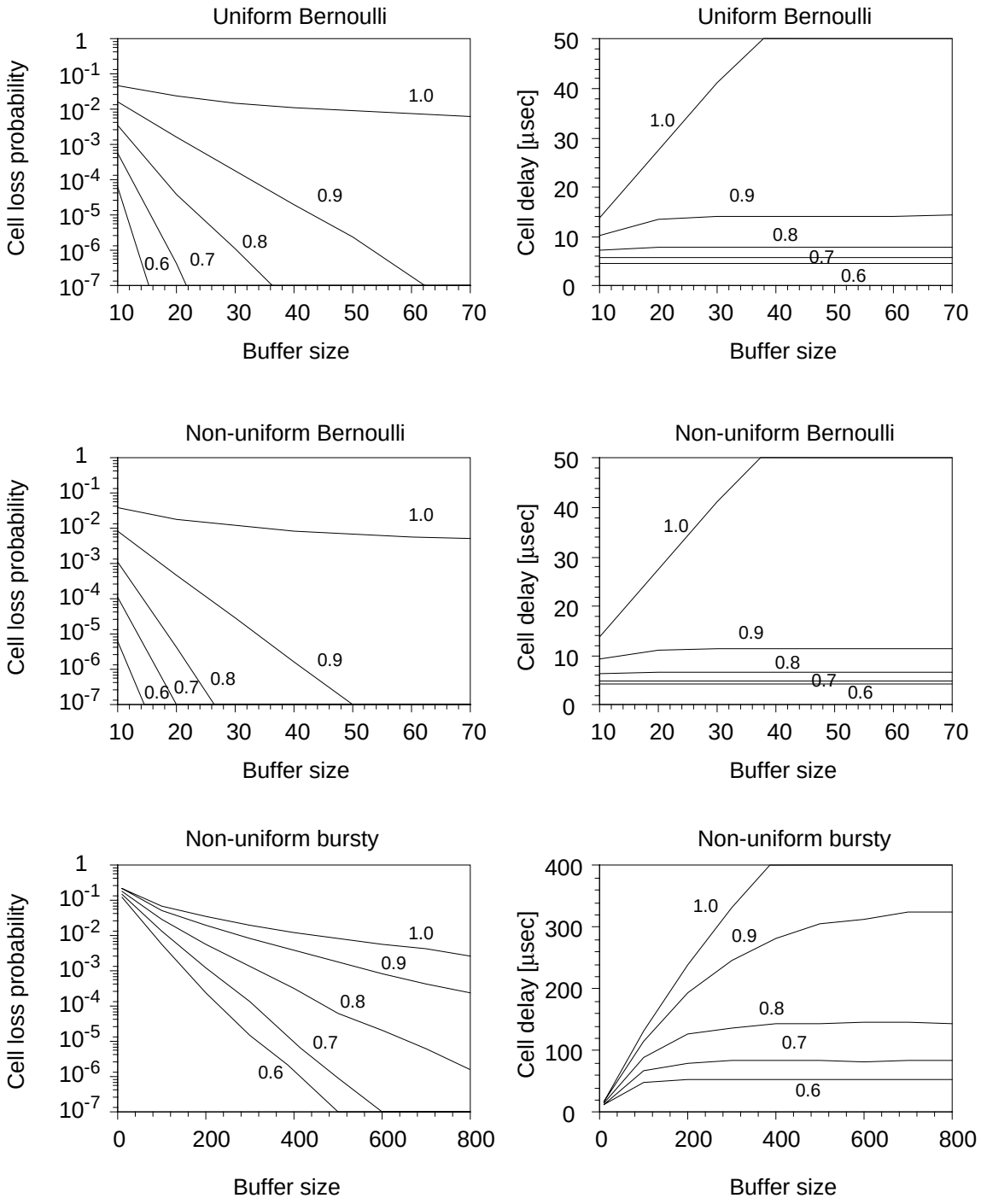


Figure 6.2: Performance of a 16 port knockout switch. with $L = 8$

We can see that the cell loss probability decreases as the buffer size increases and the smaller the offered load the smaller is the cell loss. For uniform Bernoulli traffic a cell loss probability less than 10^{-6} can be achieved for up to offered load of 0.9 with output buffers that have fewer than 60 slots. The results for non-uniform Bernoulli traffic show slightly lower cell loss than the ones for uniform Bernoulli traffic. This is due to the fact that for the non-uniform traffic we can only have up to 4 cells arrive at an output port during the same cell cycle, because of the non-uniform connection setup. This reduces the overflow probability for the output buffer. Figure 6.2 shows that for the bursty traffic there is an increase of more than an order of magnitude in the buffer slots needed to obtain similar cell loss probability as for the Bernoulli traffic.

The cell delay for uniform and non-uniform Bernoulli traffic is very similar. The cell delay increases slightly when the buffer size is small and is increasing, but then the delay remains constant as the buffer size increases for all but maximum offered load of 1.0. The delay is fairly small between 5-15 μsec, for offered loads from 0.6-0.9 respectively, due to small buffer sizes and low occupancy. For the bursty traffic, the cell delay is much higher due to larger output buffers and larger difference in delay occurs for the various loads offered. We can see that once the cell loss probability goes below 10^{-3} the effect of increasing the buffer size does not increase the cell delay since the cell occupancy does not increase much, and the delay curves flatten out.

The performance of the Knockout switch represents the performance of an ideal output buffered switching system because it approaches the best performance that is possible to achieve for the particular buffer sizes.

6.3.2. The Tandem banyan switch

The Tandem Banyan network [56] is an example of an unbuffered switch that uses deflection routing. It consists of *k* banyan networks arranged one after the other as shown in Figure 6.3. The

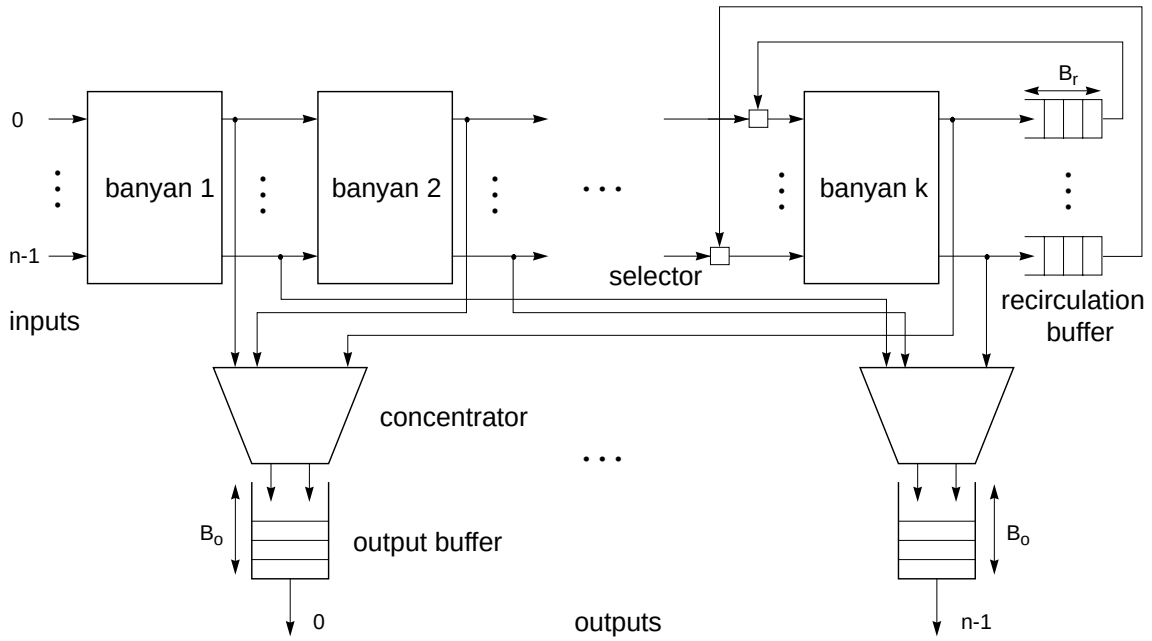


Figure 6.3: Tandem banyan switch.

outputs of each banyan are connected to the next banyan and to an output buffer associated with an output port. Each cell is routed through the banyan networks towards the requested destination output. When conflicts occur between cells that require the same output of a switch element, one cell goes to the desired output but the other is marked as misrouted and deflected off the right path. When a cell reaches an output of the banyan network, it is either sent to the associated output buffer, or if it has not been deflected it is unmarked and sent to the next banyan network where it will contend again for the output port. The last banyan network can either discard marked cells, or be connected to a set of recirculation buffers which feed back to the inputs of the last network through a selector, or send flow control signals back through the network. Delay elements ensure that cells arrive at the output buffer at the same time in correct order, however recirculation can cause cells to arrive out of order. The performance of the Tandem banyan switch depends on *k*, the number of banyan networks in tandem, the size of the recirculation buffer, B_r, and the size of the output buffer, B_o.

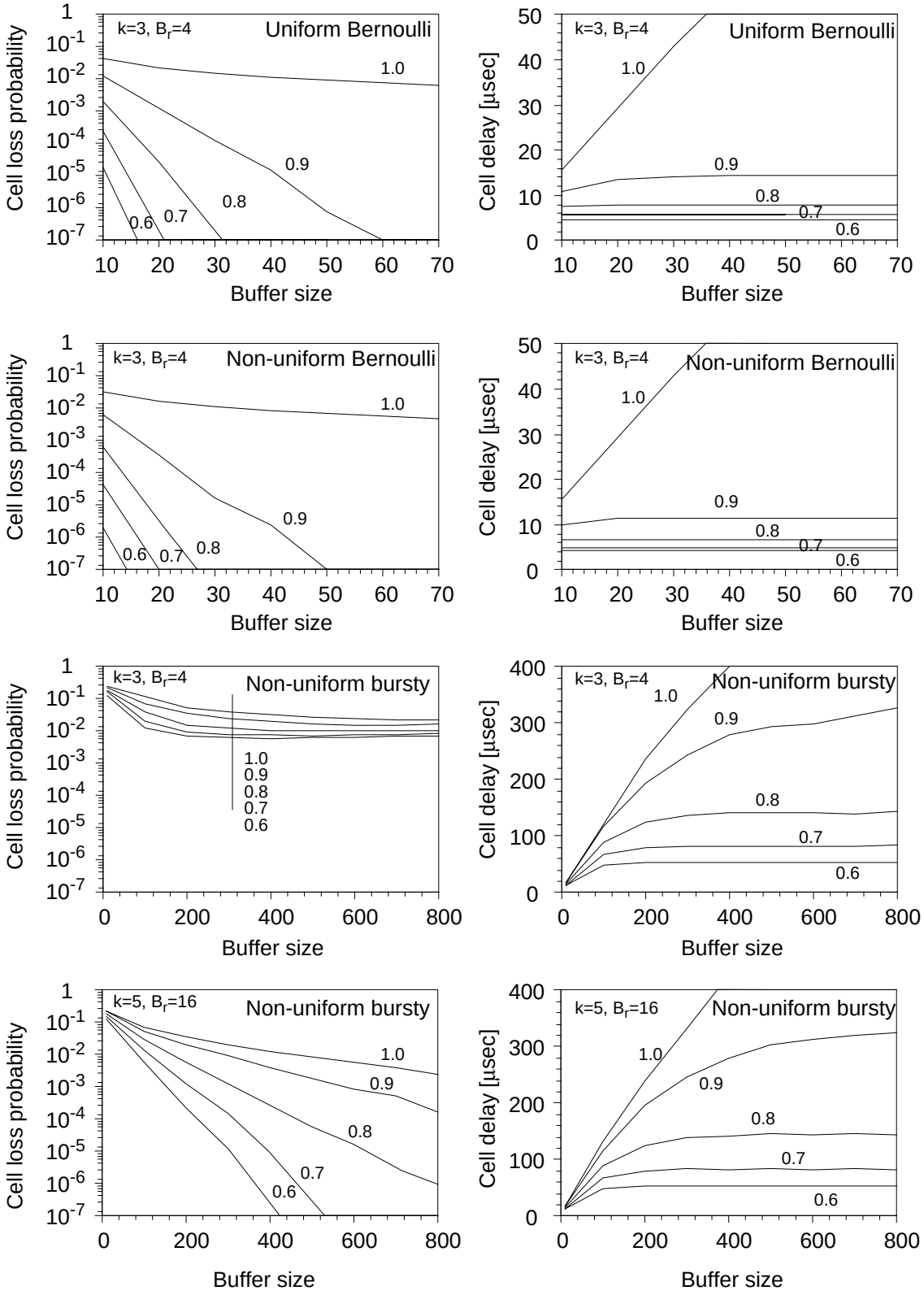


Figure 6.4: Performance of a 16 port tandem banyan switch.

Figure 6.4 shows the performance of a 16 port Tandem banyan switch as a function of output buffer size for the different traffic models. We first observe the results in the top three rows where the number of banyan networks used was $k = 4$ and the recirculation buffers have $B_r = 4$ slots each. The organization of the graphs is the same as in Figure 6.2 for the performance of the Knockout switch. In the left column we show the cell loss probability and in the right is the cell delay, both as a function of buffer size. The top, second and third rows show results for uniform Bernoulli, non-uniform Bernoulli, and non-uniform bursty traffic respectfully. The results for the Bernoulli cases are very similar to the Knockout switch with cell loss probability less than 10^{-6} for a small buffer size of 60 slots for up to offered load of 0.9. The delay is less than 15μsec for the same offered load. The performance of the non-uniform Bernoulli is slightly better than the uniform one because the probability that we have a large number of cells destined to the same output buffer is less.

The results for the non-uniform bursty traffic, in the third row, show that the cell loss probability decreases a little when we increase the buffer size but quickly levels off around 10^{-2} and stays constant for 200-800 buffer slots. The reason for this is that the loss is occurring in the recirculation buffer so no matter how many output buffer slots we add the performance will not improve. The loss in the recirculation buffers is due to the fact that the recirculation buffers are too small and more importantly the number of banyan networks is not sufficient to support the bursty traffic.

The last row shows the performance for a switch with 5 banyan networks in tandem and 16 slot recirculation buffers. In this case the cell loss probability decreases drastically. Most of the loss in the recirculation buffers has been eliminated resulting in a cell loss probability close to an ideal output buffered switch. The delay for the bursty traffic is similar to the delay for the Knockout switch since most of the delay is in the output buffer.

6.3.3. The Sunshine switch

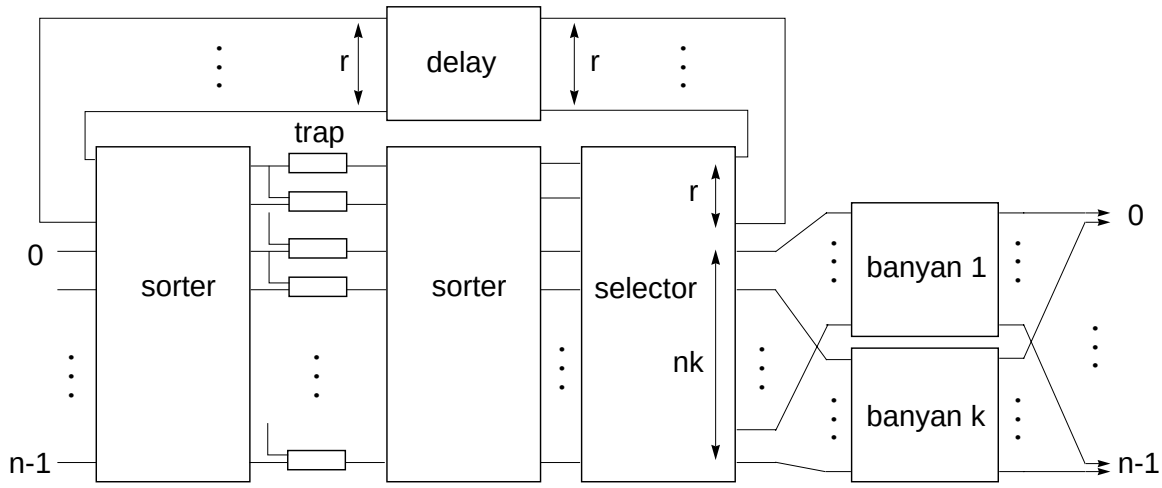


Figure 6.5: Sunshine switch.

The Sunshine switch [27] shown in Figure 6.5 is an example of a sorter based switching system. In this network a sorter is used to sort the cells by priority and destination address. A trap considers the sorted cells and marks *k* destinations to be winners, where *k* is the number of banyan networks, while the extra cells are marked for recirculation. A second sorting network separates the winners and then sorts the losers by priority. A selector routes the highest priority losers to the recirculation ports and passes the winners to the routing network which guides cells to the outputs. Multiple banyan networks are used to allow more than one cell destined for the same output to go through the network per cycle. This reduces the number of recirculation ports needed. The performance of the Sunshine switch depends on the number of recirculation ports *r*, the number of banyan networks at the output, *k*, and the number of output buffer slots, B_o.

Figure 6.6 shows the performance of a 16 port Sunshine switch with 8 recirculation ports. The organization of the graphs is similar as before. The graphs in the top three rows show results from a switch with two banyan networks and the bottom one has three banyan networks in order to

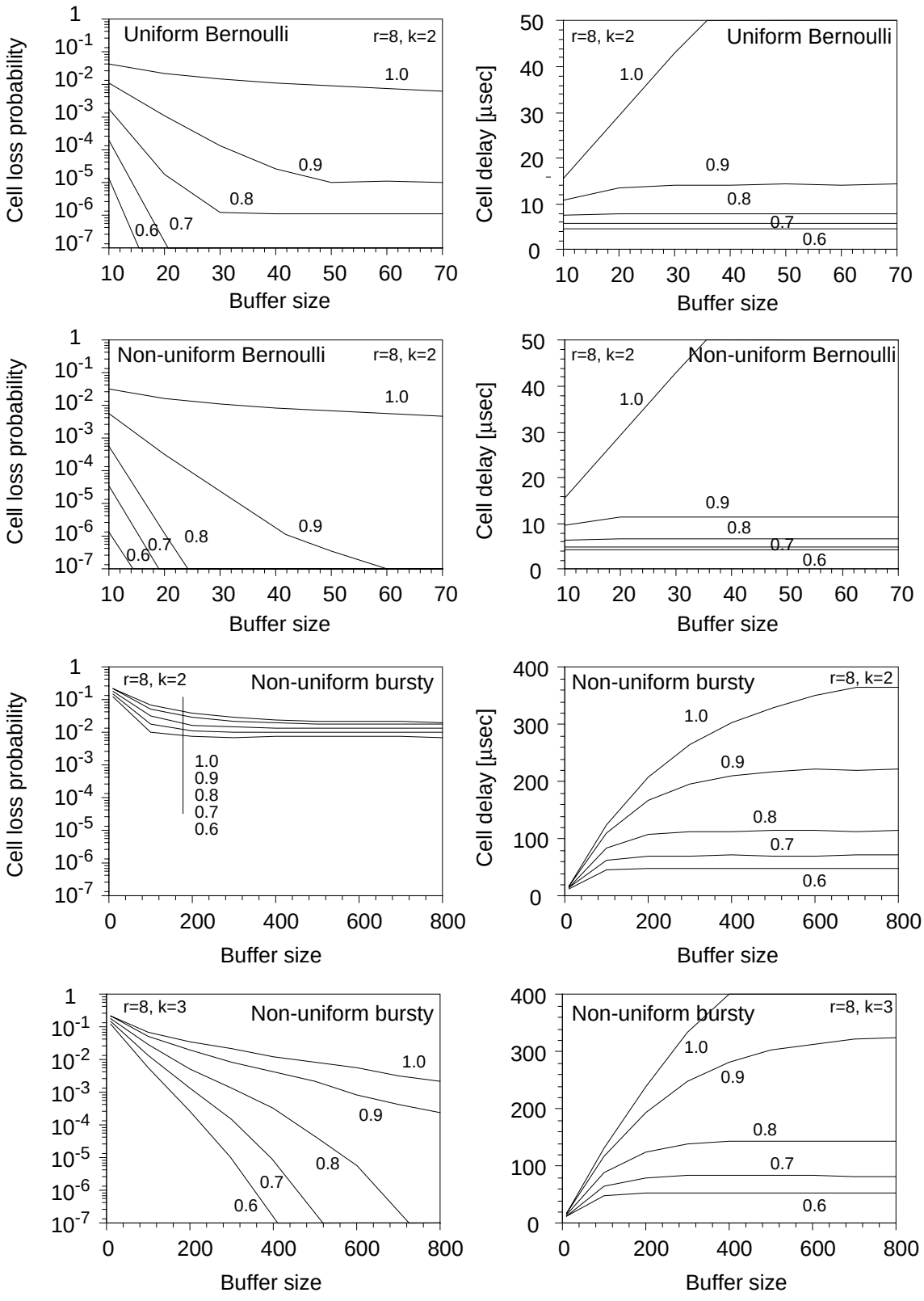


Figure 6.6: Performance of a 16 port sunshine switch.

improve the performance. Each row shows cell loss probability (left) and cell delay (right) for the different traffic models.

The cell loss probability for the uniform Bernoulli traffic (top left) shows the cell loss decrease as the buffer size increases. For offered load of 0.8 and 0.9 the cell loss probability levels off at around 10^{-6} and 10^{-5} respectively. This is due to the fact that the selector has a cell loss that becomes the limiting factor, which cell loss can not be improved upon for the given switch parameters. The number of recirculation ports or the number of banyan networks is not sufficient to support smaller cell loss. The switch with non-uniform Bernoulli traffic performs a little better and does not show loss in the selector. This is due to the fact that the probability that a large number of cells is destined to the same output buffer is lower. The cell delay for the Bernoulli traffic is similar to the delay obtained for the Knockout and Tandem banyan switches and is between 5-15μsec delay for offered loads of 0.6 - 0.9 respectively.

The results for the non-uniform bursty traffic show that the parameters *r* and *k* of the switch used for the Bernoulli traffic are not sufficiently large for bursty traffic. The third row uses the same number of banyan as the Bernoulli traffic, but it turns out that it is not sufficient for the bursty traffic and a large cell loss is observed in the switch. Again the loss is occurring in the selector as before but it is much higher. For all the offered loads the loss is about 10^{-2}. Due to the high cell loss the output occupancy decreases and the cell delay is less than the delay for the Knockout and the Tandem banyan switch under bursty traffic. The bottom graph shows the performance of the Sunshine switch with three banyan networks at the output. This is sufficient to bring the cell loss to the same levels as an ideal output buffered switch. The cell delay increases compared with the previous case due to the increased output buffer occupancy.

6.3.4. Lee's switch

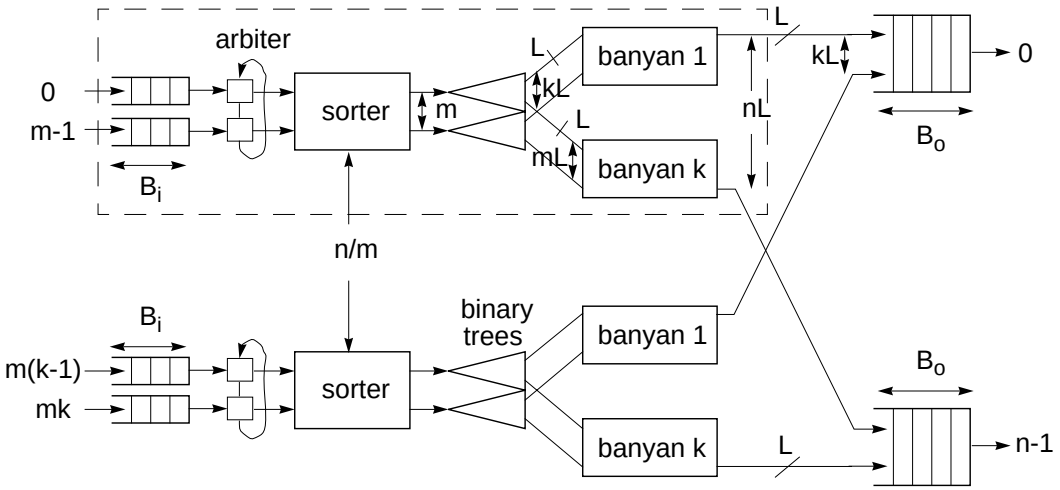


Figure 6.7: Lee's switch

Lee's switch [38] is a unification of the sorting based switching systems and the Knockout switch. The design is modular and is constructed from k independently operated switch modules that are connected by concentrators at the output buffers as shown in Figure 6.7. Each module has $m = n/k$ inputs and nL outputs. It consists of a ring arbiter that ensures that only L cells are destined to the same output port, and a sorter followed by binary trees with each having kL leaves. The binary trees are then connected to k banyan mL-networks. Each switch module has L output lines to each output buffer where a concentrator collects all the lines from the k switch modules. The performance of the switch depends on k, the number of switch modules and banyan networks in each switch module, L the parallel data path to each output port, and the number of input buffer slots, B_i, and output buffer slots, B_o.

Figure 6.8 shows the performance of Lee's switch for the three traffic pattern discussed in the previous sections. The number of switch modules used is $k = 4$ and the number of parallel paths are $L = 2$. To ensure that the input buffering does not limit the performance we choose the input

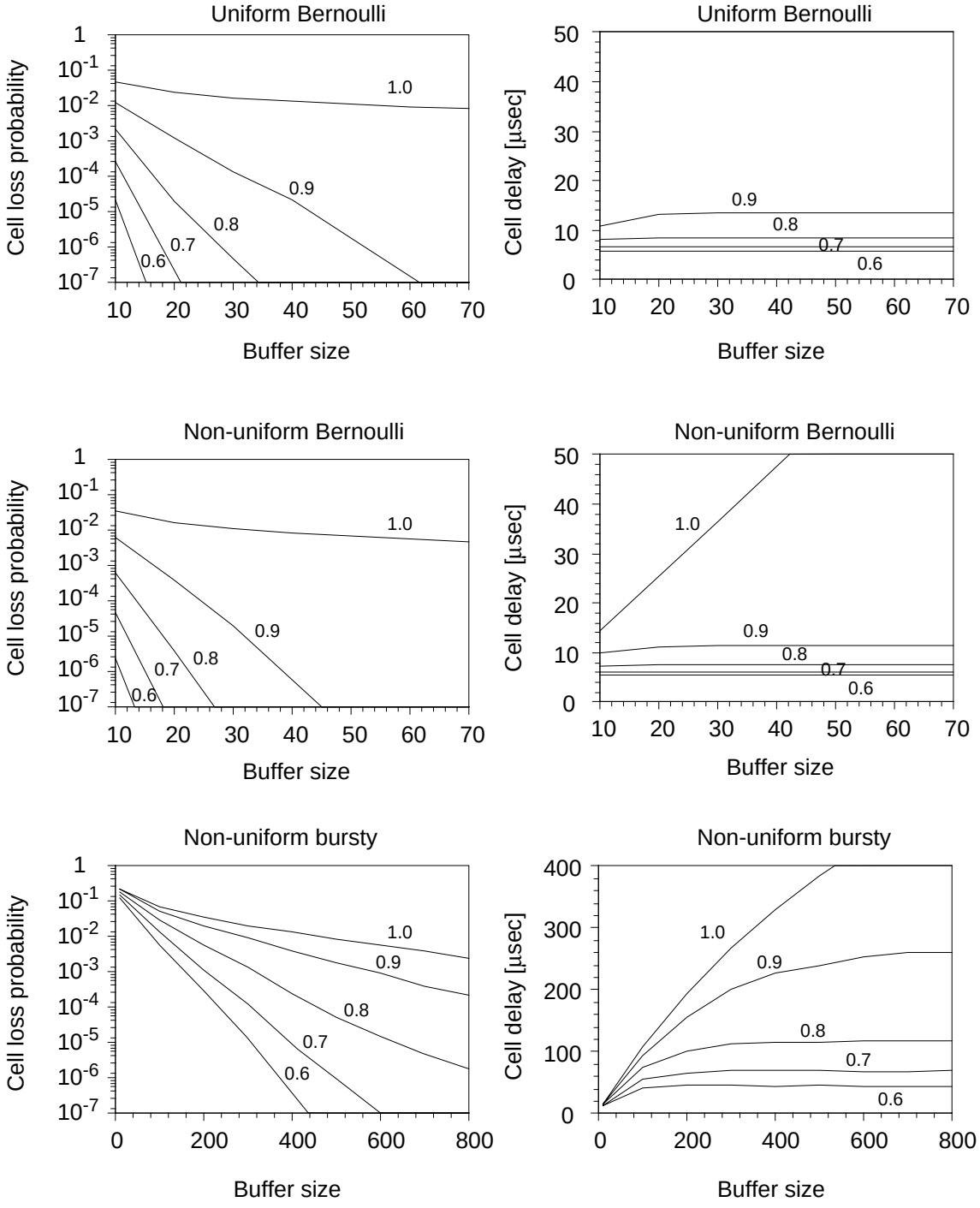


Figure 6.8: Performance of a 16 port Lee's switch with $k = 4$, $L = 2$, and $B_i = 32$

buffer to have B_i = 32 slots. The graphs Figure 6.8 show cell loss probability in the left column and cell delay in the right column. The three rows show results for uniform Bernoulli, non-uniform Bernoulli and non-uniform bursty traffic from top to bottom respectively. The results for uniform Bernoulli show that we can obtain cell loss of 10^{-6} with offered load of 0.9 for reasonably small output buffer size of 55 slots. Offered load smaller than 0.9 requires smaller buffer sizes. The cell delay is also low, between 5-15μsec, for offered loads of 0.6-0.9 respectively. In the case of offered load of 1.0 the delay (not shown) is larger than 50μsec because the input buffer fills up and causes additional delay. In all the other cases for loads less than 1.0 there is low input buffer occupancy and no loss at the input buffer. The switch performs slightly better with non-uniform Bernoulli traffic reducing the output buffer size required to obtain loss of 10^{-6} to about 40 slots for offered load of 0.9 The delay is similar to the delay for the uniform case and is about 5-12μsec for offered loads of 0.6-0.9 respectively, and fairly constant as the buffer sizes increase. The offered load of 1.0 does not cause input buffer overflow in the case of the non-uniform Bernoulli traffic so the cell delay for it is much less than in the uniform case.

Lee's switch can support the bursty traffic with the same size switch parameters *k*, *m*, and *L* as for the Bernoulli traffic. The cell loss probability is similar to the one obtained for the Knockout switch, supporting a cell loss requirements of 10^{-6} for an offered load of 0.8 with around 800 buffer slots, which is about as good as an ideal output buffered switch would perform.

6.3.5. The Benes switch

The last switch architecture we consider is the dynamic routing Benes network with shared buffer switch elements. The network is constructed from *d* x *d* switch elements that are connected in Benes topology which is described in Chapter 2. Each switch element has some buffering which is shared among all its ports. An *n* port Benes network constructed from *d* x *d* switch elements has

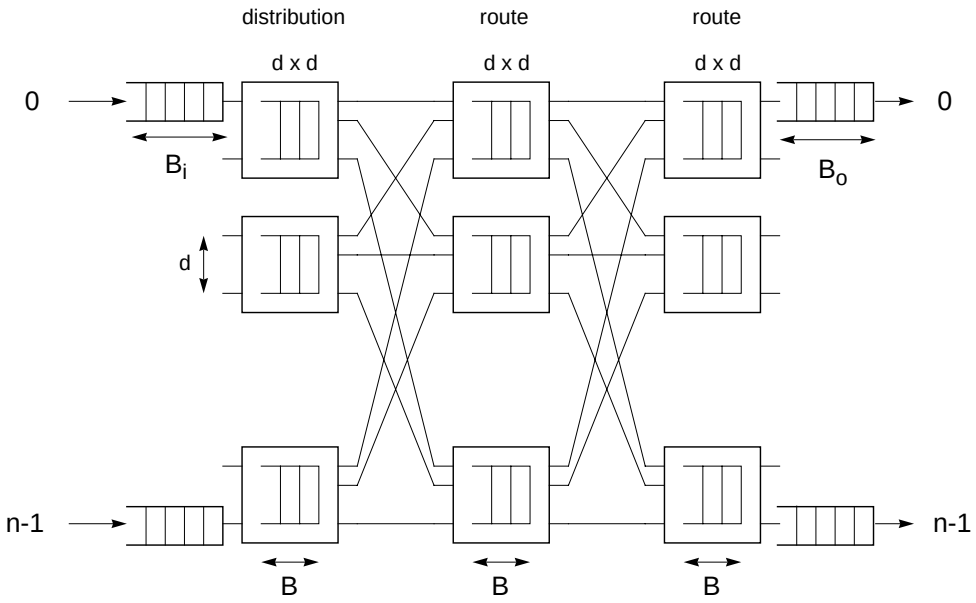
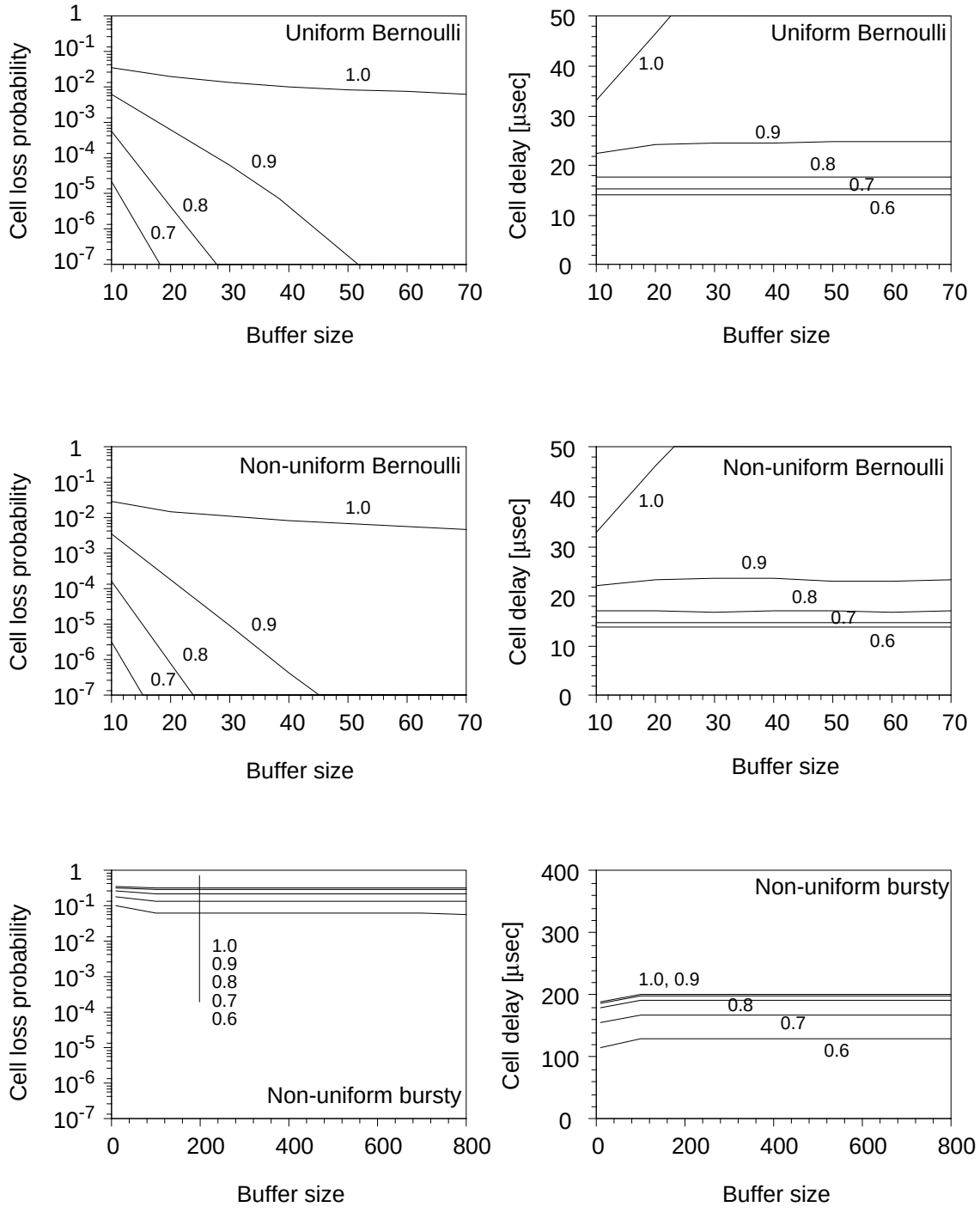


Figure 6.9: 3-stage shared buffer Benes switch

$2\log_d(n) - 1$ stages with n/d switch elements per stage. The first $\log_d(n) - 1$ stages distribute received cells randomly among the output ports for load balancing. The last $\log_d(n)$ stages route the cells to their destination ports. When more than one cell is destined to the same switch element port, only one of them can be transferred downstream while the others are held in the switch element's buffer. A grant based flow control is used between different stages in the network to ensure that the buffers in the switch elements do not overflow. Therefore cell loss can only occur at the input buffer or at the output buffer.The performance of the switch depends on the switch element dimension, d, switch buffer size, B, the input buffer size, B_i and the output buffer size, B_o.

Figure 6.10 shows the performance of three stage 16 port Benes switch. Each switch element has four input/output ports with 16-slot shared buffers. The switch has input buffer size of 32 and speed advantage of 1.25. The figure shows results for cell loss probability (left column) and cell delay (right column) for the three traffic models. The top, middle and bottom row show the results for uniform Bernoulli, non-uniform Bernoulli and bursty traffic respectively. We can see that overall the cell loss for the uniform Bernoulli case is a little lower than we have previously observed.

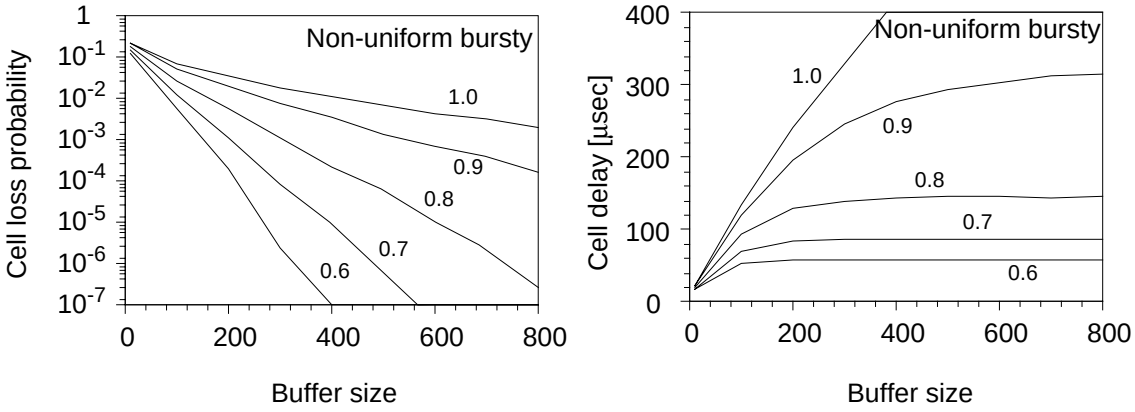


igure 6.10: Performance of a 16 port Benes switch with $d = 4$, $B = 16$, $B_i = 32$ and speedup of 1.25

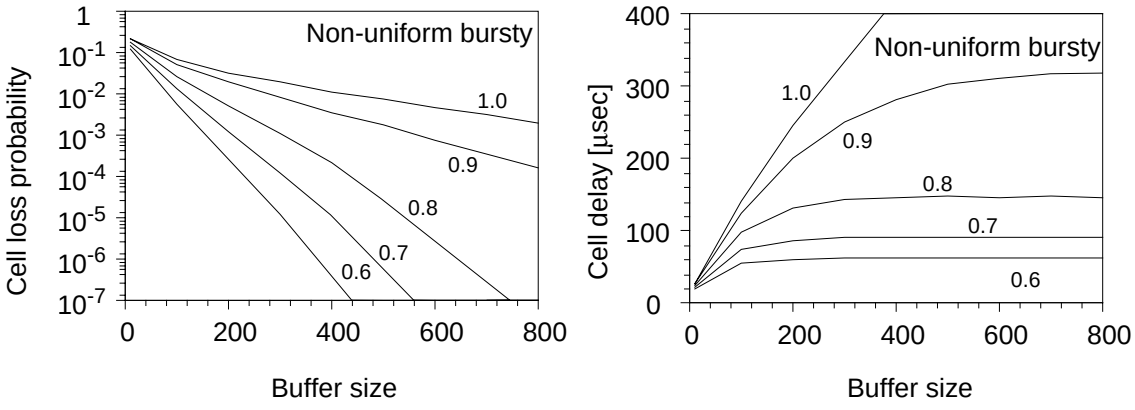
Cell loss of 10^{-6} can be obtained with buffer size of 45 for offered load of 0.9. This is due to the additional buffering inside the Benes network and at the input of the network. As a consequence we can see that the cell delay is higher than for the other switches, about 15-25 μsec for offered loads of 0.6-0.9 respectively, compared with 5-15 μsec for the same load range as seen before. The cell delay remains constant for increasing buffer size except for offered load of 1.0 which causes buffer overflow and increases delay linearly with buffer size.

Using non-uniform Bernoulli traffic results in even less cell loss probability than in the uniform traffic requiring a buffer size of 40 to obtain cell loss probability less than 10^{-6} for offered load of 0.9. This is as before, due to the fact that in the non-uniform case fewer cells destined to the same output port arrive at the same time. The cell delay for the non-uniform Bernoulli traffic is similar to the results for the uniform traffic.

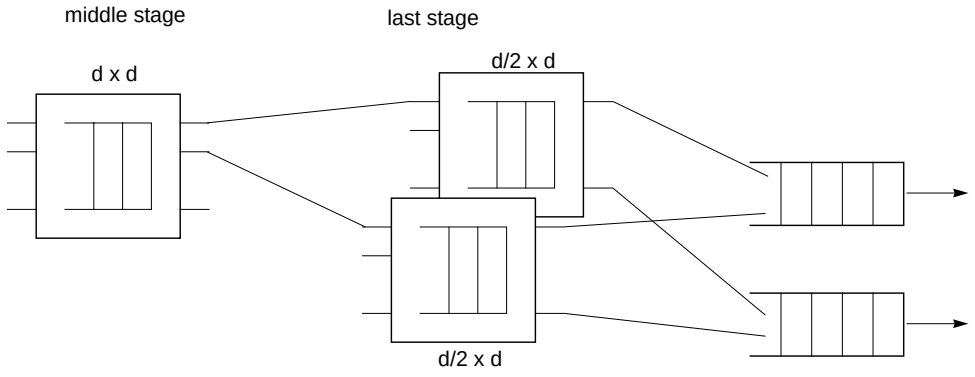
The results for bursty traffic are shown in the bottom row of Figure 6.10. The cell loss is very high, more than 10^{-1}, for all the offered loads except for load of 0.6 which has slightly lower cell loss. The cell loss remains constant when the buffer size increases, due to the fact the network is congested causing the output buffer occupancy to remain low and the input buffers to overflow and large numbers of cells to be lost at the input, independently of the output buffer size. In both Bernoulli cases the cell loss is occurring due to overflow of the output buffer, without any cell loss at the input buffer, which suggests that the speed advantage used is sufficient. In the case of bursty traffic the speed advantage is not sufficient to support the traffic anymore as can be seen from the high cell loss. The cell delay is similar for all the offered loads, due to the low output buffer occupancy, and the delay remains constant as the buffer size increases. In general, by increasing the speed advantage we improve the performance of the switching system. Figure 6.10 (a) shows the performance of the shared buffered Benes switch with the same buffering as before, and with speedup of 2.5 compared with the external links. The traffic model used in these simulations is



(a) N = 16, d =4,B = 16, B_i = 32, speedup = 2.5



(b) N = 16, d = 4,B = 16, B_i = 32, speedup = 1.5, dual switch



(c)

Figure 6.11: Performance of a 16 port Benes switch with additional speed advantage.

non-uniform bursty traffic. As can be seen in the left graph the cell loss probability is much lower than before for speedup of 1.25 (Figure 6.10 bottom left). The switch can support cell loss probability less than 10^{-6} with buffer size of 800 and offered load of 0.8. It performs as well as an ideal output buffer switch. The cell delay has increased compared with the previous speedup of 1.25, because now the output buffer has higher occupancy especially when the offered load is high. The cell delay is similar to the cell delay of the other switch architectures because of the large output buffer which causes most of the delay to occur and thus the additional shared buffers in the Benes network do not add much to the delay.

The speed advantage can be achieved by increasing the clock rate, by increasing all data paths, or by using multiple Benes networks in parallel. A less costly method to obtain close to equivalent speed advantage is shown in Figure 6.10 (c). In this network we have replaced each switch element in the last stage of the Benes network with two switch elements that both are connected to the same output buffers. Each output buffer can therefore receive two cells at every cell cycle reducing the output port conflict at the last stage. Figure 6.10 (b) shows the performance of a network that employs this method and in addition to the dual switches in the last stage it has speed advantage of 1.5. All the buffering and dimensions are the same as before. The results are again obtained for a non-uniform bursty traffic. As we can see, the switch performs very similarly to the switch with speed advantage of 2.5. Both types achieve performance close to the performance of an ideal output buffer switch. For a large network the use of a dual switch in the last stage could yield a significant reduction in complexity, compared with the complexity required for a Benes network with higher speed advantage.

6.4. Comparison

We have shown that the different switch architectures all can achieve performance that is close to the performance of an ideal output buffer switch with the right switch parameters. The Tandem banyan, Sunshine and Benes switches are more sensitive to the traffic model used than the Knock-out and Lee's switches. These traffic-sensitive switches must be engineered with respect to the more conservative bursty traffic model. This requires an additional complexity for these networks, due to the larger switch parameters.

Table 6.1 : Parameters for 16 port switching systems.

Switch	Bernoulli	Bursty
Knockout	L = 8	L = 8
Tandem banyan	k = 4, B_r = 4	k = 5, B_r = 16
Sunshine	r = 0.5N, k = 2	r = 0.5N, k= 3
Lee's	k = 4, L = 2	k = 4, L = 2
Benes	d = 4, B = 16, s = 1.25, T = 500000	d = 4, B = 16, s = 2.0, T = 500000

Table 6.2 shows an estimate of the complexity of the switching architectures in terms of the number of VLSI packages, assuming data pin limitation of 64 and clock rate of 100MHz. The equations used to calculate the numbers are from Zegura [70]. The parameters used to obtain the required performance are summarized in Table 6.1. T denotes the maximum transistor count per chip. We see that for Bernoulli traffic the Tandem banyan and the Benes network have the lowest complexity but additional complexity is needed for these networks when using bursty traffic. The Knockout switch remains with the same complexity for both traffic models and surprisingly has the lowest complexity for bursty traffic. Note though that Zegura has shown that the complexity of

the Knockout switch grows much faster than the others as we increase switch size. Furthermore the complexity of the Benes network could also be improved by using a larger switch element size. The bursty traffic causes increases in complexity for the Tandem banyan, Sunshine, and the Benes switches. For the 620Mb/s links the increase in complexity is 96% for the Tandem banyan, 15% for the Sunshine and 27% for the Benes. In the case of 2.4Gb/s links the increase is 90% for the Tandem banyan, 13% for the Sunshine and 48% for the Benes network.

.

Table 6.2 : Number of VLSI packages with pin limit 64, and clock 100MHz for 16 port network.

Switch	Bernoulli	Bursty	Bernoulli	Bursty
	620Mb/s	620Mb/s	2.4Gb/s	2.4Gb/s
Knockout	28	28	86	86
Tandem banyan	24	47	73	138
Sunshine	41	47	138	156
Lee’s	44	44	150	150
Benes	26	33	58	86

We also want to estimate the complexity of a 256 port switching systems by assuming that the parameters needed to achieve the required performance are mostly dependent on the traffic and scale according to Table 6.3. We can then calculate the complexity for 256 port system, with larger pin limit of 256 and clock of 100Mhz. Table 6.4 summarizes the complexity of the various systems. As before the bursty traffic causes increases in complexity for the Tandem banyan, Sunshine, and the Benes switches. For the 620Mb/s links the increase in complexity is 86% for the Tandem banyan, 6% for the Sunshine and 100% for the Benes. In the case of 2.4Gb/s links the increase is 82% for the Tandem banyan, 10% for the Sunshine and 176% for the Benes network. Even though the increase in complexity for the Benes network is very high the total complexity of the. Benes

switching system is significantly lower than all the other systems. The Knockout switch has now the largest complexity, which was expected for large network sizes.

Table 6.3 : Parameters for 256 port switching systems.

Switch	Bernoulli	Bursty
Knockout	L = 8	L = 8
Tandem banyan	k = 5, B_r = 4	k = 7, B_r = 16
Sunshine	r = 0.5N, k = 2	r = 0.5N, k= 3
Lee's	k = 4, L = 2	k = 4, L = 2
Benes	d = 16, B = 48, s = 1.25, T = 500000	d = 16, B = 48, s = 2.0, T = 500000

Table 6.4 : Number of VLSI packages with pin limit 256, and clock 100MHz for 256 port network.

Switch	Bernoulli	Bursty	Bernoulli	Bursty
	620Mb/s	620Mb/s	2.4Gb/s	2.4Gb/s
Knockout	1792	1792	6144	6144
Tandem banyan	260	484	928	1696
Sunshine	456	484	1429	1568
Lee's	260	260	928	928
Benes	117	234	150	414

For bursty traffic the large output buffers may also become a large factor in the complexity measure, this factor is not taken into account in the numbers of Table 6.2 and Table 6.4 since the buffering requirements are similar for all the switching architectures.

6.5. Remarks

This chapter compared the performance of several popular switch architectures using simulation. The comparison was made with exactly the same traffic models for all architectures for a fair comparison. It was shown that all the architectures can achieve performance similar to an ideal output buffered switch with the right switch parameters. The Knockout and Lee's switch do not require any additional hardware complexity to support the bursty traffic beside the additional output buffers, to obtain similar performance as for the Bernoulli traffic. The Tandem banyan, the Sunshine and the Benes switch all require additional hardware complexity to support the same performance using bursty traffic as for the Bernoulli traffic. Without the additional switch complexity the cell loss probability of the switches is very high. The additional complexity varies from 6-15% for Sunshine and 46-176% for the Benes switch and around 90% for the Tandem banyan switch depending on the link rate. For the 16 port systems the Tandem banyan and the Benes switch have the lowest complexity when using Bernoulli traffic but the Knockout switch has lower complexity when considering bursty traffic. For the 256 port systems the Knockout has significantly higher complexity than the other systems, and the Benes system had the lowest.

Bursty traffic will require additional complexity compared with Bernoulli traffic for networks larger than the ones simulated in this chapter. It would be interesting to simulate their performance to obtain better estimate how much increase in complexity is for the different traffic models for larger networks and if it affects the relative complexity of the switching architectures as these results show.

In this chapter we have also demonstrated the flexibility and the modelling capabilities of the simulation tool, by modeling and simulating the performance of these widely different switch architectures.

7. TRANSIENT TRAFFIC BEHAVIOR IN SWITCHING SYSTEMS

Thus far in this thesis we have considered long term or steady state performance of switching systems. In this chapter we will examine the transient traffic behavior of a switching system using a shared buffered Benes switching network. The focus will be on congestion conditions and congestion clearance. The goal is to examine how long congestion periods last in such a switching system after the input traffic load has been limited, and how the length of such periods can be reduced. Furthermore we look at what effects a local congestion can have on the whole switching system, and how the effects can be decreased.

The organization of the chapter is as follows. We first discuss related work and then we describe the simulation studies and the results. We divide the studies into two parts. The first part is on congestion clearance after overloading the system with Bernoulli and bursty background traffic. The second one examines the congestion effect on the switching system when two bursts destined for the same output port collide. The chapter ends with a summary.

7.1. Related Work

The transient study is possible through interactive use of the simulation tool presented in Chapter 5. Since such tools have not been available to researchers in the past, studies like this one have been difficult to perform. Therefore there has not been much work on transient traffic behavior in switching systems and most studies have focused on steady state averages.

Soung C. Liew[39] has done a study on the performance of input and output buffered ATM switch design principles under uniform and bursty traffic. In his simulation study he examines the effect of different buffering strategies, and the effect of the speed advantage of the switching network and the output link speed on cell loss probability. He found that unless buffers are shared or

are very large, strategies that improve throughput under uniform random traffic are not very effective under bursty traffic. Furthermore, he showed that many qualitative results true for uniform random traffic are not true for bursty traffic.

Recently Logothetis and Trivedi[40] did a transient analysis of the leaky bucket rate control scheme under Poisson and bursty traffic. They give a mathematical model for the transient behavior and suggest that the traditional steady state measures might not be appropriate for today's networks because a network connection is of finite duration and steady-state might never be reached. Short term metrics such as cell delays or cell-loss within blocks might be more appropriate.

7.2. Congestion Clearance

In this section we want to examine the time it takes for congestion to clear up in a shared buffered Benes switching network. In order to do that, we first put the network in traffic overload conditions for a long period by offering the maximum possible load to the system. The switching system is therefore filled with cells which are located at the various buffers. We then change the traffic to some lower level of background traffic and examine the network as it emerges from overload and the congestion clears up. We particularly observe the total number of cells in the network as a function of time and the delay of cells exiting from the network. Furthermore we study the buffer occupancy in the switching system. The network has some number of stages of switch elements each with shared buffering. We divide the buffers in several categories: the input buffers, buffers that are in the same stage of the switching network, and the output buffers. Then we inspect the instantaneous occupancy averaged for each of the buffer categories. The switching network used for the simulations is a 64 input Benes network using 8 port switch elements, each with buffering for 32 cells shared among all the ports as shown in Figure 7.1. The network has therefore 3 stages, identified as first, middle and last stages. The system also has a 32 cell input buffer at

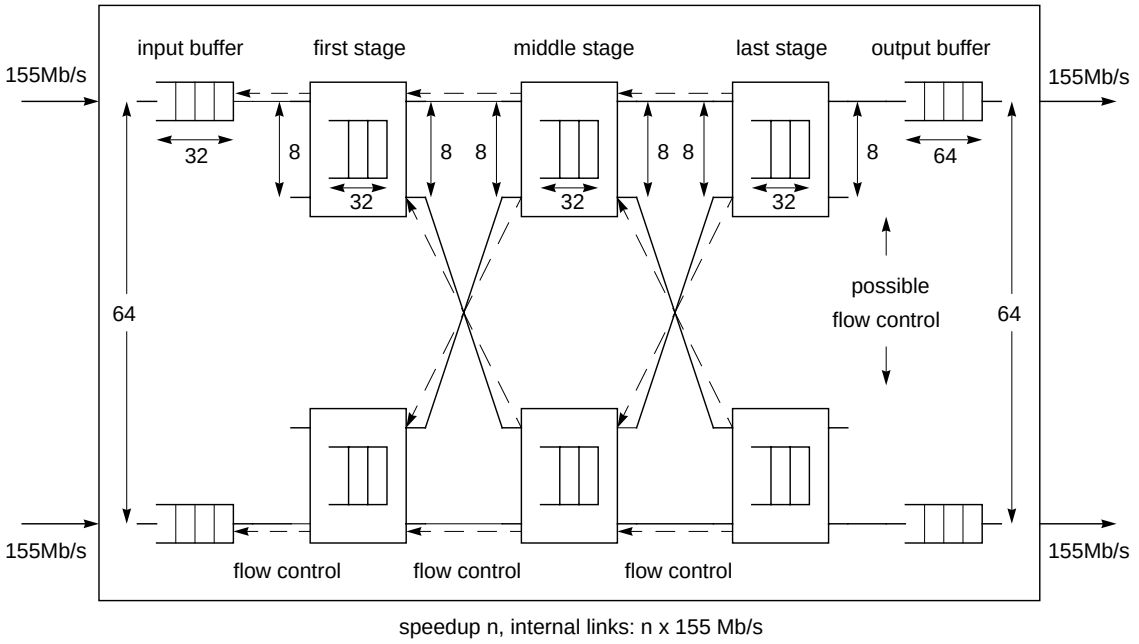


Figure 7.1: 64-port shared buffered Benes switching system.

each input port and a 64 cell output buffer at each output port. The external links are assumed to have bandwidth of 155 Mb/s but the switching system can operate faster internally giving it a speed advantage referred to as the speedup factor.

We look at two types of switching networks. The first type has a flow control between stages in the switching network and also between the output buffers and the switching network in order to have no internal cell loss and limit the cell loss to the input side only. The other network type has only flow control between stages in the switching network and can therefore have cell loss both at the input and the output side.

In the next two subsections we discuss results obtained from studying the congestion clearance after overloading with Bernoulli and bursty background traffic.

7.2.1. Bernoulli background traffic

We first examine the congestion clearance by using Bernoulli background traffic after we reduce the offered load from the maximum possible. The results obtained for a network with output buffer flow control are summarized in Figure 7.2 The figure has two columns. The left column has results for a network without speed advantage and the right one has results for a network with internal speed that is twice the external link rate. There are three set of panels in each figure.

The two plots in the top row show the total number of cells in the systems as a function of time. Each graph in the top panel has four curves, for different values of background traffic. All the curves show that the number of cells decreases as a function of time from the overload condition to some small value, and this decrease is faster the less background traffic is present. If we compare the two plots we see that the decrease is faster for low background load in the left but is faster in the right for high background load. This is due to difference in where the cells are located for the networks with and without speed advantage. Furthermore, the steady state number of cells is larger with higher background loads.

The panel in the middle plot the delay that the cells exiting the system encounter as a function of time. Again each graph in the panel has four curves for the different background load. The delay shows similar behavior as the plots showing the total number of cells. The delay drops as a function of time and the drop is faster for small background traffic than for large ones. The delay starts out flat at around 150 μsec for awhile but then drops down to between 10-20 μsec in all cases. The delay for no speed advantage is less than the delay for the network with speed advantage except for background load of 0.7. This is as before due to the difference in occupancy of the output buffer where most of the delay for the second network is occurring. The time for the delay to drop is from 200 μsec. for background load of 0.1 to about 800 μsec. for background load of 0.7.

The bottom panel plot the buffer occupancy averaged over the different categories mentioned earlier for a particular background load, $\rho = 0.5$. Each graph in the panel shows five curves, one for each of the categories of buffering. From these curves we can determine if there was congestion in the switching network and where the cells are located in the system. In both the graphs we can see that there is congestion in the switching system at the beginning. The input buffer occupancy starts falling when the load is reduced to the background level and once it empties, the occupancy in the first stage of the network falls quickly followed closely by the middle stage of the network and finally the last stage. Note that the last stage of the network has fewer cells than the first and middle stages. The network without speed advantage has very little cell occupancy in the output buffers because the network has less capacity than the output links. The network with the speed advantage has much higher occupancy in the output buffer, but surprisingly it is not full in the beginning. The reason for this is because of the output buffer flow control. When one output buffer overflows the cells start to back up in the network and cause congestion in the middle stage for cells going to other outputs. We will examine this further in a later section. The network with speedup of 2 has congestion in the network for less than 200 μsec but the one without speed advantage remains congested for about 350 μsec.

The results obtained for a network without output buffer flow control are summarized in Figure 7.3. The figure is organized as Figure 7.2. The results are similar but with some important differences. If we compare Figure 7.3 to figure Figure 7.2 we see that the networks with no speed advantage (left column) have almost identical results in spite of the fact that the networks of Figure 7.3 do not have output buffer flow control. This is due to the fact that the congestion occurring is not due to the output buffer overflowing but rather output contention in the middle and last stages of the network which causes the first stage and then the input buffers to overflow. When comparing the networks with speed advantage (right column of Figure 7.2 and Figure 7.3) we see

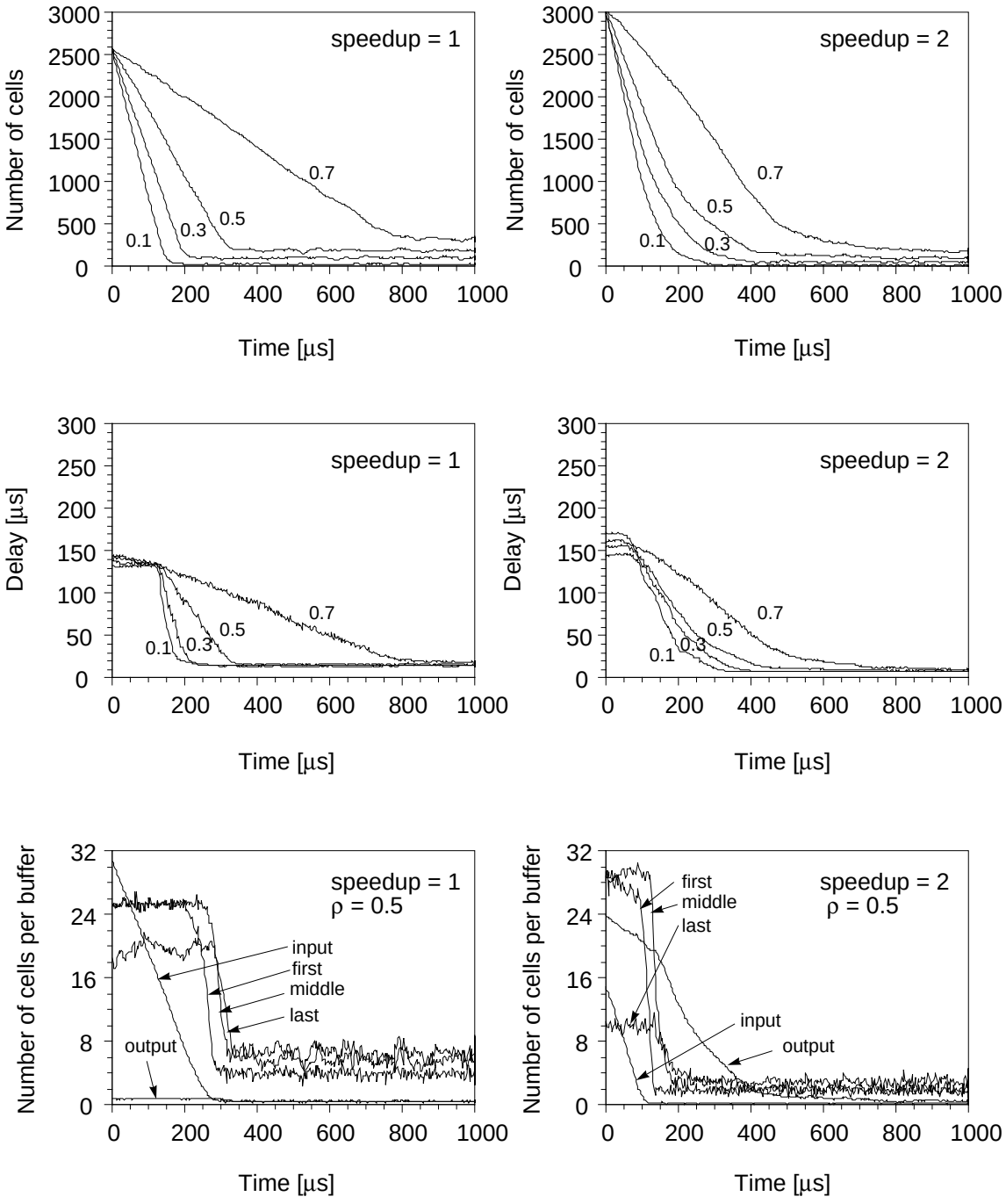
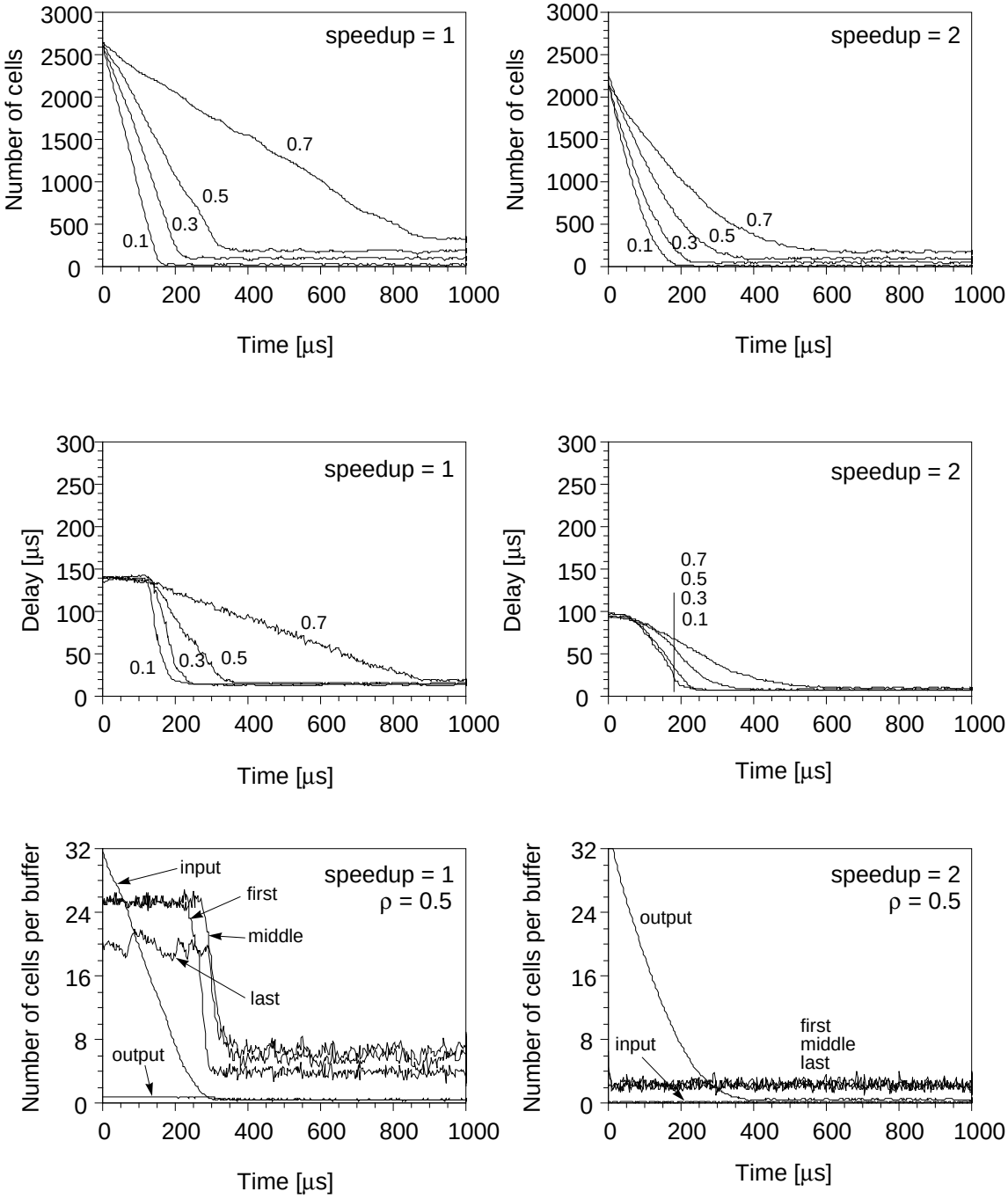


Figure 7.2: Congestion clearance for network with output buffer flow control and Bernoulli background traffic.



igure 7.3: Congestion clearance for network without output buffer flow control and Bernoulli background traffic.

that for the network without output buffer flow control (Figure 7.3) the number of cells in the network is smaller and almost all the cells are in the output buffer with no congestion inside the switching network and no buildup at the input buffer. The number of cells decreases faster and even for high background load the number of cells drop to steady state in about 500 μsec compared with 700 μsec with output buffer flow control. The delay seen by the exiting cells is less than before starting at 100 μsec and dropping faster even for higher background load.This is because cells encounter very small delay in the switching network and almost all the delay is in the output buffer.

It is interesting to note that the maximum number of cells is less than 3000 even though the buffer capacity of the switching system is 6912 buffer slots. This suggests that the buffers are not used very efficiently, especially the output buffers. For the network with speed advantage the input buffers are hardly used and most of the buffering should be moved to the outputs.

7.2.2. Bursty background traffic

We now examine congestion clearance for bursty background traffic after we reduce the offered load from the maximum possible. Figure 7.4 and Figure 7.5 show the results obtained. All the results are for a network without output buffer flow control. The organization of the panels is the same as for the Bernoulli traffic.

Bursty traffic is modelled as in the previous chapter as a two state active/idle source with the time in each state geometrically distributed. The cells are generated at full rate during the active period. The burst length used is 20 cells long. The whole burst is assigned the same destination port. The average rate of the source for each simulation is shown next to each curve.

Figure 7.4 shows that using bursty background traffic sources results in much slower congestion clearance than when using Bernoulli background traffic (Figure 7.3). For a network without

speed advantage (left column) the congestion does not clear except for very low background load of 0.1 and 0.3. From the top panel we can see that the number of cells in the system reduces more for the lower background load than the high background load. For the network without speed advantage the number of cells does not reduce at all for load of 0.7 suggesting that the network cannot support the load and therefore the congestion does not clear up. For the network with speed advantage the number of cells in the system is reduced but the higher the background load the higher is the number of cells in the steady state. The total number of buffered cells during overload is also higher than for the Bernoulli traffic, at around 3500-4000 cells compared with 2300 cells. This is mostly due to increase in the input buffer occupancy and congestion in the switching network. The delay (middle panel) has increased considerably and remains high for speedup of one but for speedup of 2 the initial delay is around 175 μsec compared with 100 μsec for Bernoulli traffic and drops very slowly taking almost 800 μsec to reach a steady state value compared with 200-400 μsec for Bernoulli traffic.

In the bottom panel we can see that for the network without speed advantage the congestion does not clear up since both the buffers in first and middle stages remain almost full and the input buffers do not empty. This is due to the output contention in the middle stage, which in spite of the lower background load does not get resolved. The network with speedup of 2 does experience congestion in the switching network that clears up after about 400 μsec compared with no congestion in the Bernoulli case.

The congestion can be avoided if we increase the speed advantage more as done in Figure 7.5, which shows results for a network with three times the speed advantage in the left column. The delay seen by the exiting cells has decreased even more and is similar for all background loads, since most of the delay is due to output buffering. The number of cells reduces to lower levels than before, but there still is a considerable number of cells for high background loads in the steady

state. The congestion in the network has disappeared in the bottom left graph since there is little buffering in the three network stages and most of the cells are in the output buffer.

The last set of results plotted in the right column of Figure 7.5 is for a network without speed advantage compared to the external links but in the last stage with a switch element that has double the buffer size, 64 slots, and two links to each output buffer. This is an inexpensive way to increase the performance of the switching system without requiring a speed advantage of 2. The graphs show a great improvement in performance compared to the results obtained for a network without speed advantage (Figure 7.4 right column) and similar performance to a network with speed advantage of 2 (Figure 7.4 left column). When in overload condition the modified network has fewer cells and the congestion clears up faster for low background load. For higher background load the performance is not quite as good as the network with speedup of 2, due to higher output contention in the middle stage as can be seen in the bottom right graph of Figure 7.5.

7.3. Congestion effect of coincident bursts

In this section we examine congestion conditions caused by bursts when two bursts are destined for the same output port and collide in the switching system. We use the same network as in previous section, both with and without output buffer flow control.

The simulations were run until the network reached a steady state using Bernoulli traffic and uniform output destinations, with offered load of 0.5. Then at some point in time noted as time t = 0, two inputs start transmitting cells to the same output port at full maximum rate. The two sources transmitted bursts of 50, 100 or 500 cells. Once all the cells in the burst were injected into the network the sources returned to the original behavior. Depending on the length of the bursts and the speed advantage of the network, congestion may occur in the network. The congestion can last for a long time after the bursts are over.

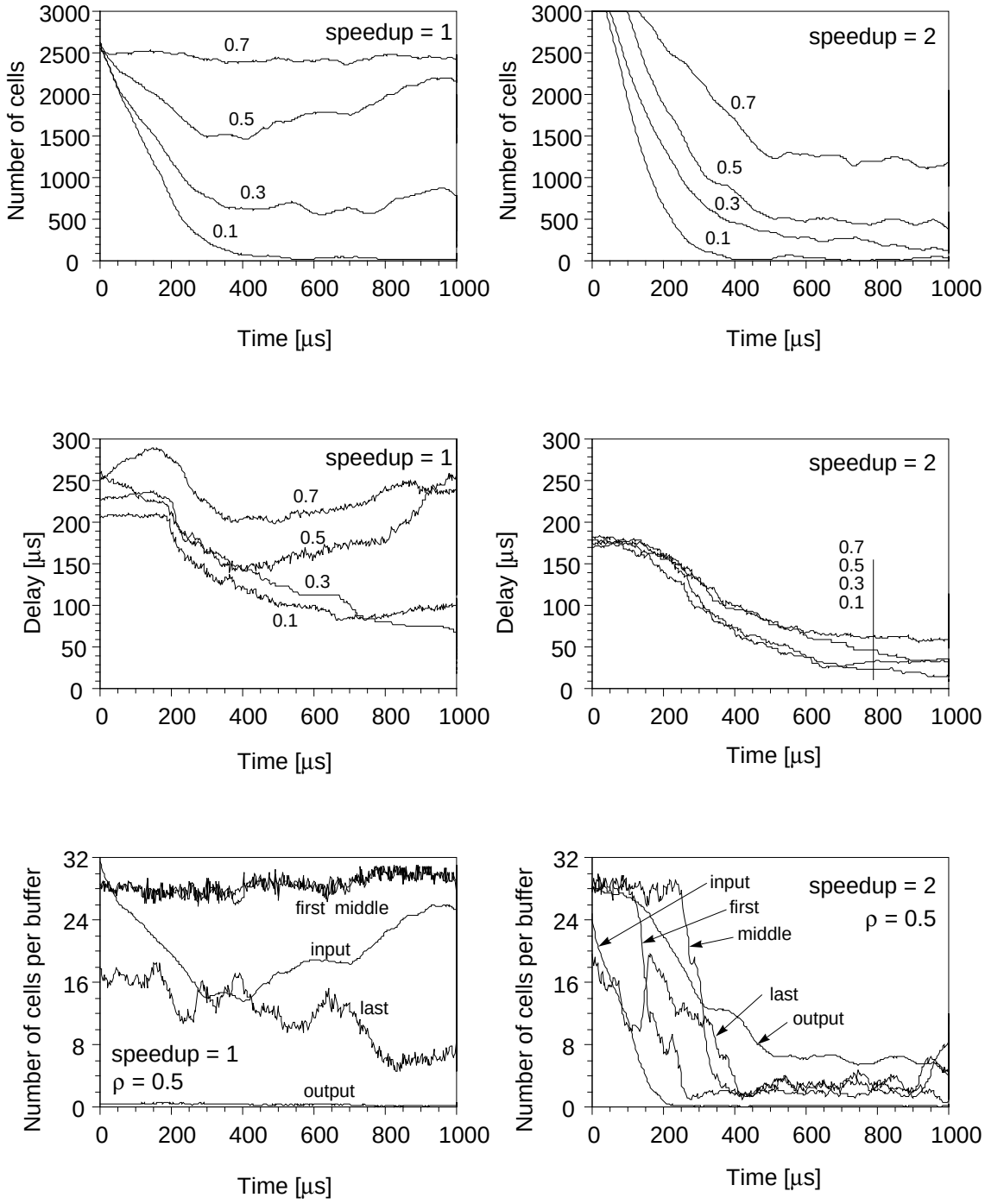
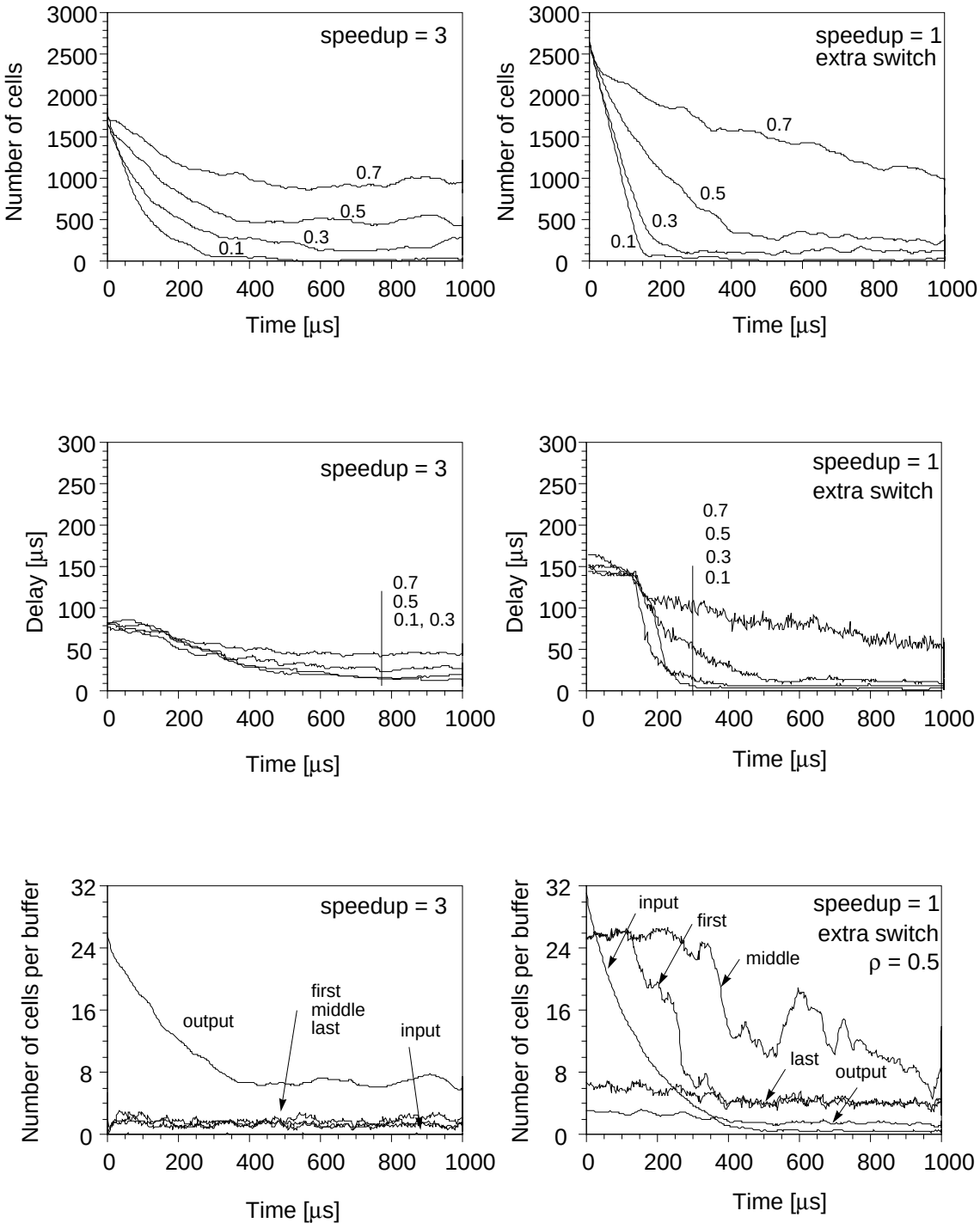


Figure 7.4: Congestion clearance for bursty background traffic without output buffer flow control



igure 7.5: Congestion clearance for bursty background traffic without output buffer flow control

Figure 7.6 shows the results for a network with output buffer flow control. There are three rows in the figures each for a different speed advantage of the network compared with the external links. Each graph plots 3 curves as a function of time, one for each burst length. The graphs in the left column plot the total number of cells in the system and the ones on the right plot the delay that the cells exiting the network encounter. The dotted lines across the graphs show the time at which the bursts arrivals end. We can see from the graphs that all the bursts cause an increase in the number of cells in the system, and that the longer the burst the higher this increase is. For burst of 500 cells the network saturates with around 2600 cells in the network. The number of cells keeps increasing even after the burst is over because of the congestion caused by the bursts and the effect of the congestion on all the other traffic. Increasing the speed advantage decreases the effect of the smaller bursts but does not have any effect on the larger 500 cell bursts. The delay of the exiting cells shows similar behavior as the number of cells. Shortly after the number of cells increases the delay increases, as we would expect to happen because of the longer queue lengths.

Figure 7.7 shows the number of cells per buffer in the different stages of a network with output buffer flow control. The left column shows the results for 100 cell burst and the right column shows the results for 500 cell bursts, both for Bernoulli background load of 0.5. Each graph has six curves. There is one for each of the five buffer categories: input buffer, first stage, middle stage, last stage and output buffer. The sixth curve, denoted buffer0, shows the occupancy for the buffer that the bursts are sent to. The graphs show that for the 100 cell bursts the congestion lasts for a shorter period when the speed advantage is increased from 1 to 3; the congestion period is 1100 μsec, 800 μsec and 700 μsec respectfully. For the 500 cell burst the congestion period is similar for all cases, between 2100 - 2300 μsec. By looking at the occupancy for the different stages and from the operation of the network we can conclude that the cells from the bursts contend for the same output buffer and fill up the output buffer and/or the last stage switching element leading to the

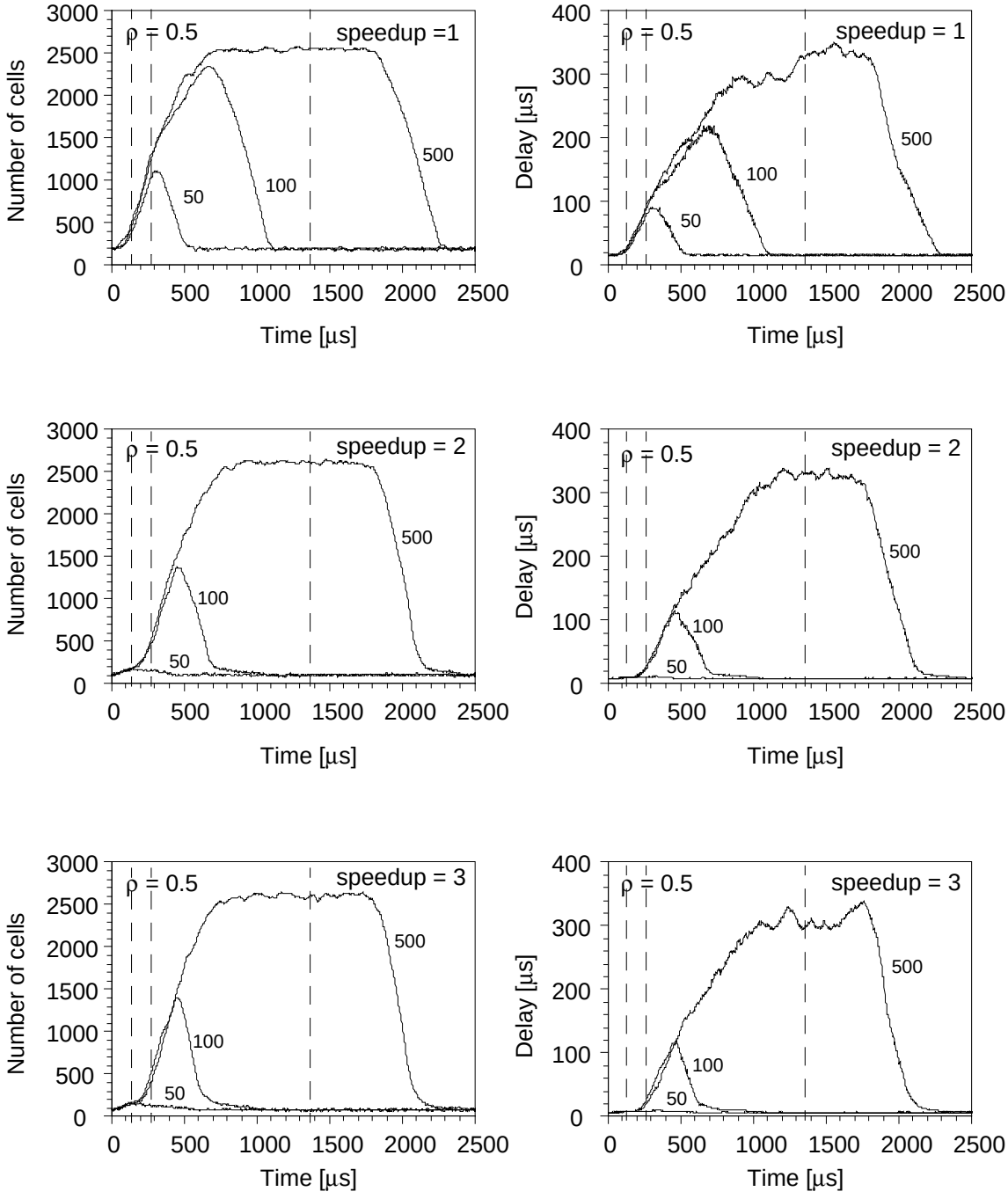


Figure 7.6: Burst congestion effect for network with output buffer flow control (cells and delay)

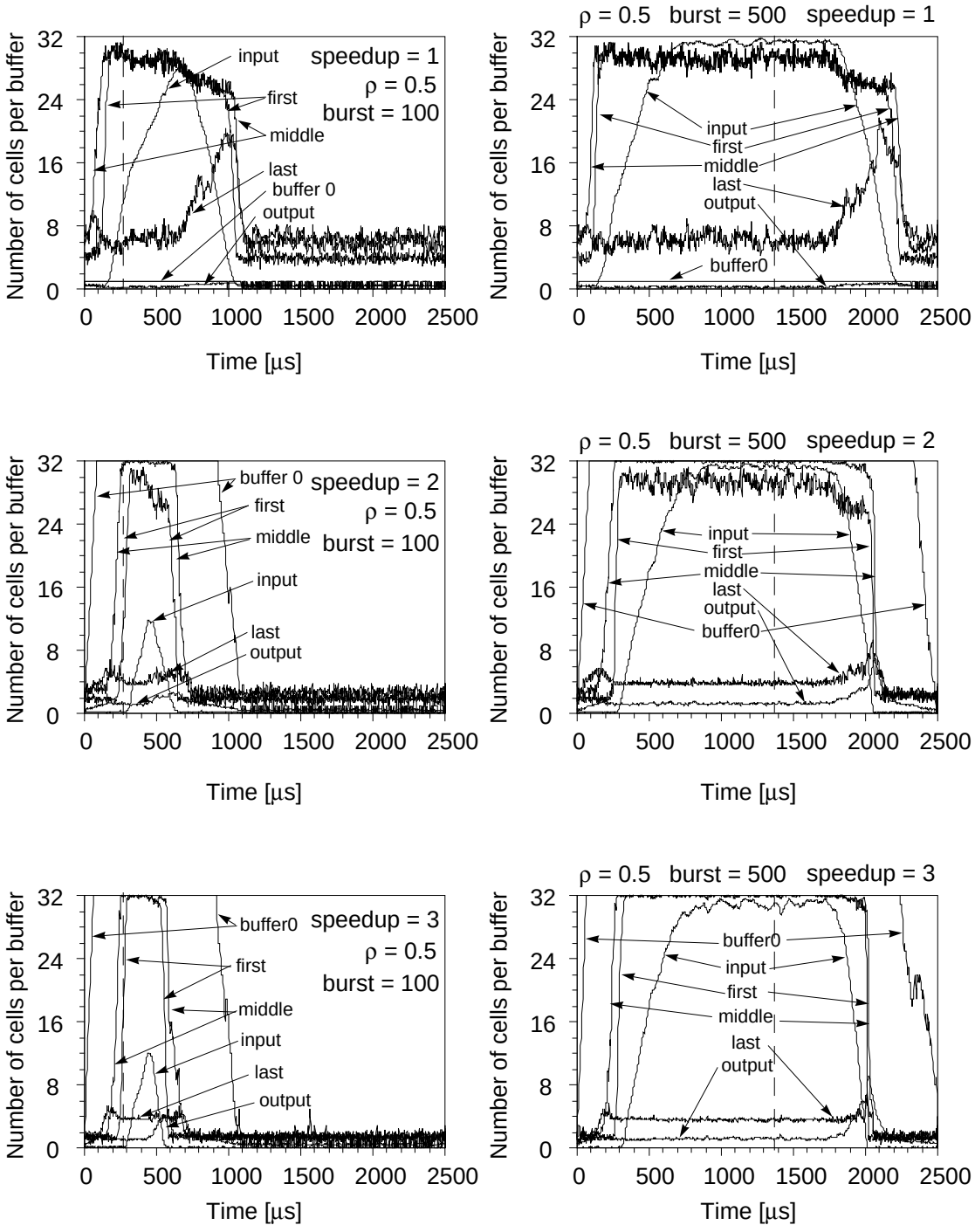


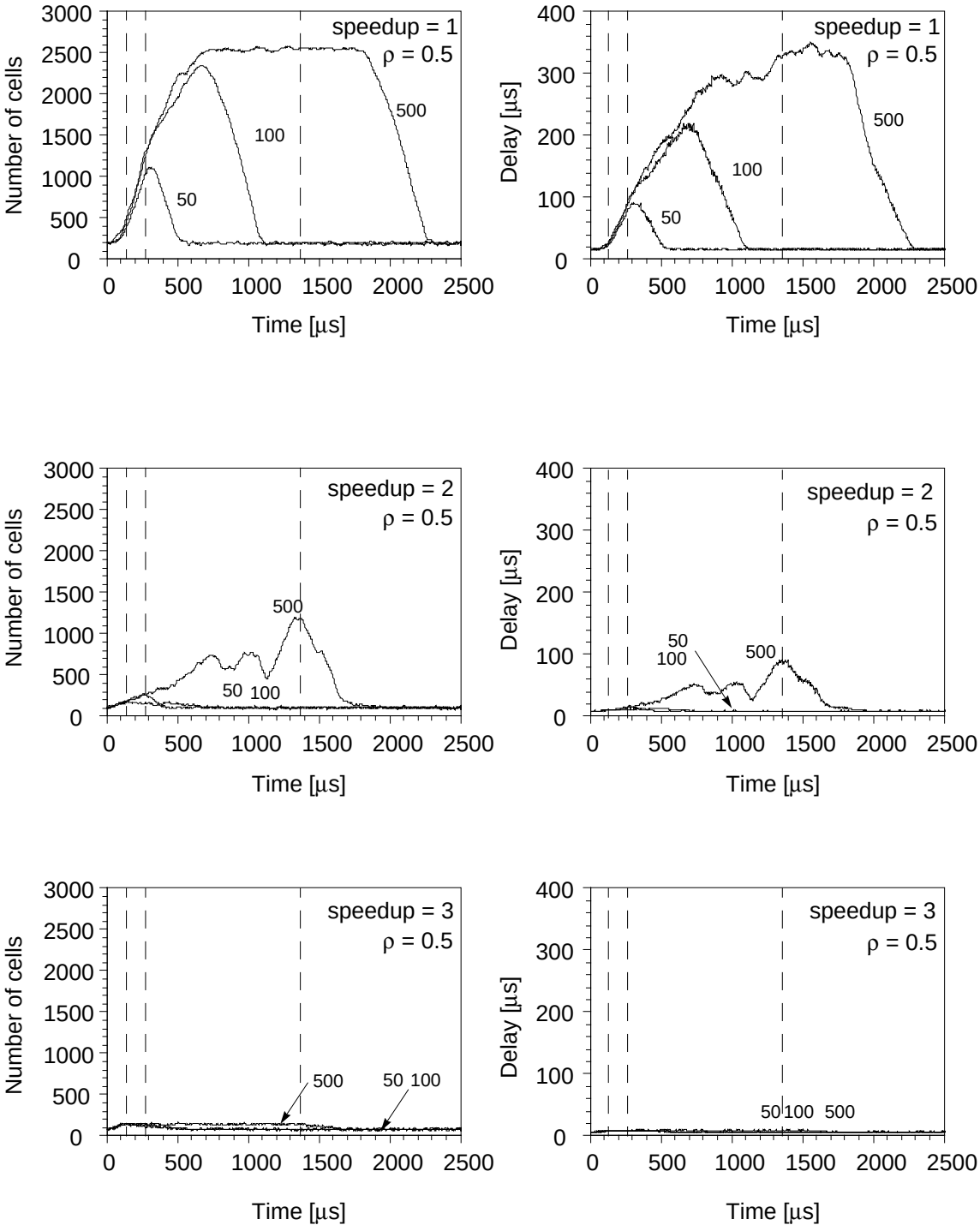
Figure 7.7: Burst congestion effect for network with output buffer flow control (buffer occupancy)

output buffer. Because of the flow control, the traffic builds up and fills up the middle stage switching elements causing congestion for the whole switch. Once the bursts are over, most of the cells in the middle stage are destined to the same last stage switching element. The congestion does not start to clear until those cells are gone and they no longer interfere with cells going to other output ports. The number of cells at the output buffers is small except for, the buffer that the bursts are sent to. Similarly, the last stage of the switching network has low occupancy since only one of them is overloaded. The main congestion is in the middle and first stage. For the larger 500 cell burst the congestion lasts long enough to fill up the first two stages and the input buffer causing the loss of a large number of cells.

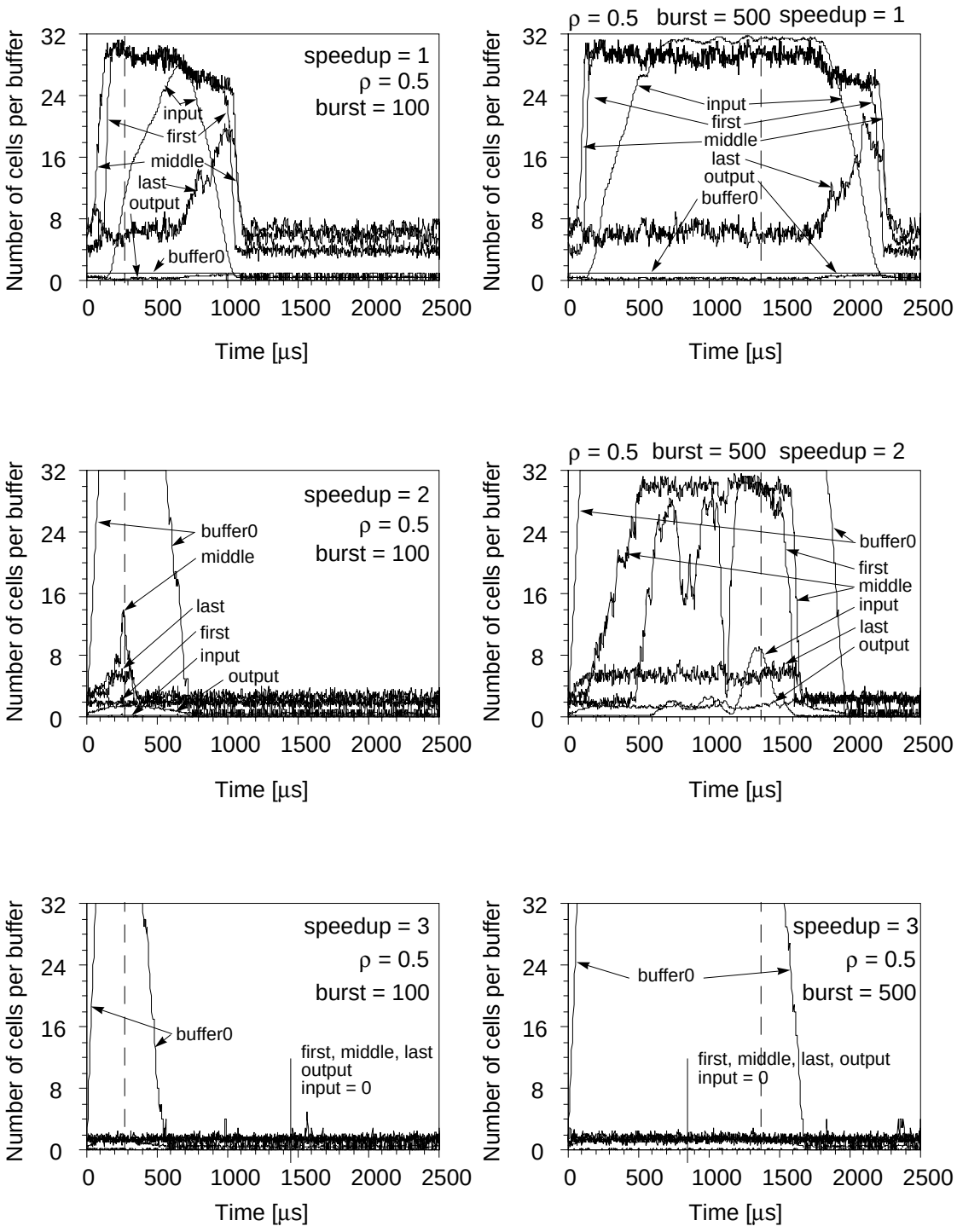
Figure 7.8 shows the results for a network without output buffer flow control. The figure is organized the same way as Figure 7.6. We can see that with speedup of 1 there is hardly any difference from the results with flow control, but when the speed advantage is increased the number of cells in the system decreases dramatically. For speed advantage of 3 the burst causes no effect at all on the network and for speed advantage of 2 only the large 500 cell burst is affected. Similarly the delay seen by the exiting cells decreases compared with the flow control networks, just as the number of cells do.

Figure 7.9 shows the buffer occupancy for the different stages in the switching system. We can see that there is only congestion in the network without a speed advantage and for the 500 cell burst in the network with speedup factor of 2. By using a speed advantage of 3 we have no congestion in the switching network. All cells in the burst are sent to the requested output port. The destination output buffer gets filled up and the additional cells are lost, but do not interfere with the other cells in the system.

For a network with speed advantage of 2 we have some interesting effects for the 500 cell bursts. The total number of cells and the cell delay oscillates during the burst period. We can see



igure 7.8: Burst congestion effect for network without output buffer flow control (cells and delay)



igure 7.9: Burst congestion effect for network without output buffer flow control (buffer occupancy)

that this is due to changes in occupancy in the input buffer and the buffers in the first stage of the network. These changes occur fairly regularly and are examined further in Figure 7.10 where the bursts are made 5000 cells long. We can see that the occupancy varies regularly during the burst period. The reason for this phenomenon are changes in the traffic distribution, which can be observed through animation of the simulation during the time period. In the beginning of the congestion the traffic backs up to the inputs and the input queues start building up. At that time the cell rate from the burst to the first stage is 1/8 of the total because of round robin service. The input buffers that the bursts arrive to keep building up, but the others start emptying because fewer cells

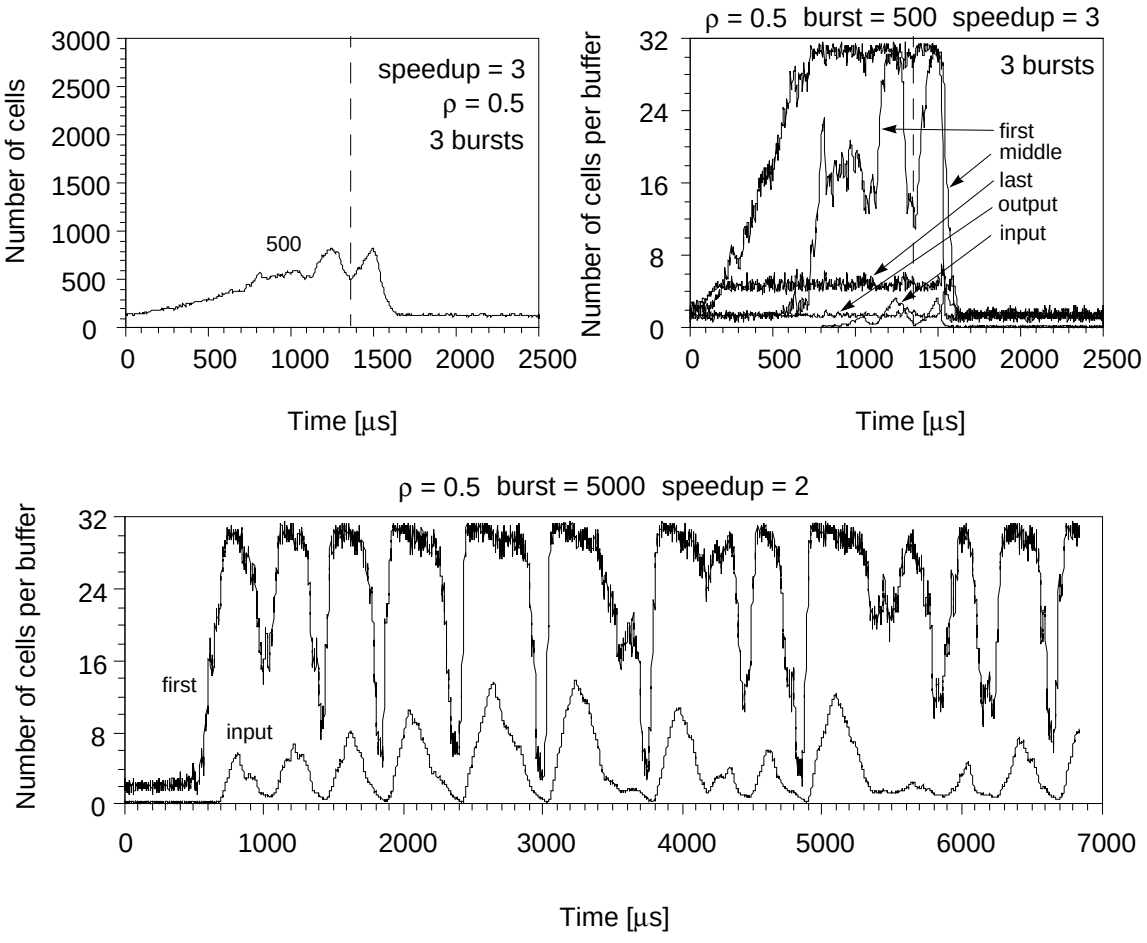


Figure 7.10: Congestion in a network with speedup of 3 and 3 bursts (top). Periodic congestion effect (bottom).

in the network are destined to the congested output buffer. Once the other input buffers are empty, the rate that the cells from the bursts enter the switch changes to close to 1/3 of the total rate of arriving traffic. This causes congestion to build up again all the way to the input buffers and the cycle repeats.

Increasing the speedup of the network compared with the external links is very effective in reducing the congestion in the network. The speedup needed is dependent on the traffic and even though a speedup of 3 is enough to avoid the congestion for two 500 cell bursts and a background traffic of 0.5, it is not enough if we have three bursts colliding as we can see in Figure 7.10. To avoid the congestion in this case we need to increase the speedup to 4, or use bandwidth allocation and call admission to reduce the probability that many cell bursts destined to the same output port collide.

7.4. Remarks

We have examined the transient traffic behavior in a shared buffered Benes network. We have found that the congestions take a long time to clear up unless the background load is very low. The congestion clearance time can be improved by increasing the speed advantage of the switching network compared with the external links. Networks with output buffer flow control have more severe congestion and take longer to clear up than those without output buffer flow control. When using bursty background traffic we found that the congestion clearance took longer than when using Bernoulli background traffic. Cells encounter higher delay, and more cells remain in the system after the congestion has cleared up. For high background load the congestion did not clear without speed advantage. For bursty background traffic a speed advantage of 3 is needed to remove internal congestion of the switching network, compared with a speed advantage of 2 for the better behaved Bernoulli traffic. We also saw that collision of two bursts can cause a severe

congestion in the switching system even long after the bursts are over. Networks with output buffer flow control are much more likely to suffer from congestion than networks without output flow control. For the flow controlled networks, increasing the speed advantage reduces the congestion time for small bursts, but for large bursts the congestion time remains the same in spite of the speedup since the output buffer overflows and cells back up in the network and cause congestion. By removing the output buffer flow control we reduce the effect of the bursts and with sufficient speed advantage we can eliminate the effect. All the cells in the bursts are sent to the destination output buffer and no congestion builds up in the networks. If the output buffer overflows, the incoming cells will get lost. From our results we conclude that even though output buffer flow control might improve cell loss for Bernoulli random traffic, such flow control should be avoided because these networks are much more prone to congestion than networks without output buffer flow control. Furthermore, additional speed advantage and extra output links can remove the internal congestion in the networks without the flow control.

We can conclude that transient traffic studies like the one presented give us a useful insight into the behavior of switching systems. Such studies also demonstrate the usefulness of the simulation tool presented in Chapter 5 that made this study possible.

8. CONCLUSIONS

The goal of this thesis is to provide tools to evaluate the performance of high speed switching systems. These tools can assist in the effective design of switching systems to support current and future communication needs. In this chapter we summarize the results of this thesis, review its contributions, and discuss directions for future work.

8.1. Queueing Analysis of Copy Networks

In Chapter 3 we considered concatenation of two networks, a popular paradigm for multicast switching. In such cascading, the first network, the copy network, performs cell replication, while the second network, the routing network, performs the routing of cells to the desired outputs. The copy networks are used to perform cell replication required for multipoint connections, and to distribute cells for load balancing. The multipoint system considered is referred to as copy-then-route, and consists of switching networks that have internal buffering.

The contributions of chapter 3 are the first proposed analytical models to evaluate the performance of copy networks The models presented provide methods for analyzing the queueing behavior of copy networks constructed from binary switches. A model for switches employing output buffering was derived and then generalized to include switches with input and shared buffering. The performance results of the copy networks obtained from the analytical models were compared with simulation results with special attention to their dependence on fanout and network size.

The results showed that the models are accurate for a single 2x2 switch element but approximate for multistage networks. The multistage analytical model overestimates the throughput, due to the assumption of independence between network stages. The models accurately reflect the

dependence of the throughput on the fanout and network sizes, show that copy networks sustain a high throughput in spite of small buffers, and act as additional buffering to the routing network. Furthermore, the throughput stays constant as the network size increases in contrast to the decreasing throughput of the routing networks.

The analysis presented can be easily extended to handle networks that use different flow control methods and non-uniform traffic and it would be interesting to see how much the accuracy of the model improves in the multistage case when there is no flow control between stages in the network. Improving the accuracy of the models for multistage networks by taking into account the dependencies of switch elements in the same stage is important. A possible improvement is to use a model extension similar to the one proposed by Atiquzzaman and Akhtar [3] in which state information is included regarding the output link that was requested when a cell was blocked at a switch element.This would take some of the intra-stage dependencies into account leading to a more accurate model.

Improving the computational performance of these models would be very useful especially by reducing the total number of iterations needed for convergence. Extending the models to allow switch elements of arbitrary dimension may also prove analytically tractable but the computational complexity is probably too high to be of practical value.

8.2. Queueing Analysis of Shared Buffer Switching Networks for Non-uniform Traffic

In chapter 4 we considered models using non-uniform traffic patterns, because uniform traffic does not represent a realistic view of traffic in a real system. Non-uniform traffic models not only reflect better the traffic patterns that need to be accommodated, but also may cause the network's performance to deteriorate to much lower levels than the ones predicted by uniform traffic analysis.

This chapter provides a method for analyzing the queueing behavior under non-uniform traffic patterns by extending the queueing analysis for buffered networks. In the analysis shared buffered switch elements are used because they have be shown to have better performance than input or output buffer elements.

Two models were presented; the vector model which is an exact model for a single stage but is very computationally intensive, and an approximate model, the active output model which reduces the complexity at the expense of accuracy. The active output model was compared with simulation on the basis of accuracy, where the performance was measured in terms of maximum throughput and probability of cell loss. The results obtained with the single stage model are exact for a 2 x 2 switch element and are close to the simulation results, both for the maximum throughput and the probability of cell loss for larger switch elements. Finally we compared the performance of multistage networks for various traffic patterns and evaluated the accuracy of the model by comparing the analytical results to simulation results. The analysis showed that the model gives a good approximation of the throughput for non-uniform traffic, and results similar to other proposed models for the uniform case.

As further work we would like to improve the computational complexity, especially the rate of convergence, so that the model can be applied to larger networks with larger switch element dimensions and buffer space. The model will most likely be more accurate for large switch elements due to the fact that the virtual output buffers become more independent as the number of outputs grows.

8.3. Switching System Simulator

The design and analysis of switching systems require the development and evaluation of extremely complex models. Simulation can be used for the evaluation of these complex systems

and can give accurate results for detailed models. However, the construction of models of large switching systems using general purpose packages can be extremely tedious and time-consuming. Furthermore, the execution speed of large simulation models built using such packages is generally too slow for effective use. This led us to the development of a simulator specific for switching simulations.

As was presented in chapter 5, we used an object-oriented approach in order to provide visual interactive simulation for network performance analysis. The visual approach provides tools that support visualization of simulation objects such as static graphics for viewing the networks and animated graphics for viewing movements of cells, snapshots of network states, and evolution of statistics. The style of visual simulation provides a new dimension of understanding unavailable in traditional simulation modeling.

The contribution of chapter 5 is twofold. First, it provides a fundamental framework for visual simulation and performance evaluation of switching systems. Second, it provides models for a variety of complex switching architectures that can be simulated in the same environment for easy comparison.

The simulation tool for switching systems developed and described in chapter 5 are needed by many. Switching system designers and researchers will be able to easily use such a tool to explore different architectural alternatives. Such a tool can also be helpful for people who would otherwise be reluctant to use simulations or performance analysis, and as a teaching aid for students. Furthermore it can be used by network managers to help them decide how to configure a network in support of particular traffic requirements.

A number of improvements could be made to the simulation tool. First it is important to make the addition of new models for network elements easier. Second, because the visualization of large networks is hard and limits the use of the visual simulation, new methods need to be developed to

convey the vast amount of information generated. Finally, a possible improvement upon the simulation's running time would be to use analytical models to generate an initial state for the simulations allowing the simulations to reach a steady state faster.

8.4. Performance of Switching Networks

In chapter 6 we evaluated and compared the performance of several well known switching systems proposed for ATM using simulation. All the systems were studied under exactly the same traffic conditions so their performance would be fair and could be easily compared. The performance was obtained both under Bernoulli traffic and bursty traffic. We determined what increase in hardware complexity was needed, besides additional output buffering, for bursty traffic in order to obtain similar performance as for Bernoulli traffic.

Simulation was used to evaluate the performance of the switching systems via the switching simulation tool described in chapter 5. This comparison study also served as a demonstration of the modeling capabilities and flexibility of the switching simulation tool by modeling and simulating the performance of widely different switch architectures.

The contribution of chapter 6 is a side by side comparison of the performance of various switching systems and the examination of the additional complexity needed for bursty traffic. It was shown that all the architectures considered can achieve performance similar to an ideal output buffered switch with the right switch parameters. The Knockout and Lee's switch do not require any additional hardware complexity to support the bursty traffic, besides the additional output buffers, in order to provide similar performance as for the Bernoulli traffic. The Tandem banyan, the Sunshine and the Benes switch all require additional hardware complexity to support the same performance for both bursty traffic and Bernoulli traffic. However, these architectures provide superior scaling properties to the Knockout, making them more economical in large configuration in spite of the added complexity.

8.5. Transient Behavior of Switching Networks

In chapter 7 we studied the transient traffic behavior of a shared buffered Benes switching network with focus on congestion conditions and congestion clearance. The goal was to examine how long congestion periods last after the input traffic load has been limited, and how to reduce the length of such periods. Furthermore, we studied the effects of local congestion on the whole switching system, and how to decrease such effects.

The contribution of chapter 7 is mainly in providing new insight into the operation and behavior of buffered switching networks.We found that congestion can take a long time to clear up unless the background load is very low. The clearance time can be improved by increasing the speed advantage of the switching network compared with the external links and by not using output buffer flow control as it results in more severe congestion and a longer clearance time. Furthermore, collision of two bursts can cause a severe congestion in the switching system even long after the bursts are over. Again networks with output buffer flow control are much more likely to suffer from congestion than networks without the output flow control, and the effect of the bursts can be reduced with sufficient speed advantage. By increasing the speed advantage we increase the loss of cells in the burst but reduce or eliminate the congestion experienced by other connections destined to other outputs.

From our results we conclude that even though output buffer flow control might improve cell loss for Bernoulli random traffic such flow control should be avoided because these networks are much more prone to congestion than networks without output buffer flow control. Furthermore, additional speed advantage and extra output links can remove the internal congestion in networks without the output flow control. It is possible that using some priority scheme would also help in reducing the congestion or the clearance time, but that needs to be studied further. Many other aspects of the transient behavior of switching networks need to be studied.

The type of study presented here gives great insight into the behavior of switching systems and shows the usefulness of the simulation tool which made this study possible.

8.6. Final Remarks

Effective design of switching systems is critical to the success of future communication systems. This thesis has provided some tools to evaluate the performance of proposed switching systems. These tools can help in finding directions to pursue in improving the existing designs.

9. ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. Jonathan Turner for all his support and guidance in the research work and in the writing of this thesis. I would also like to thank the other members of my thesis committee, Dr. Roger Chamberlain, Dr. Paul Min, Dr. Bixio Rimoldi, and Dr. Gurudata Parulkar for their constructive comments. I also thank all the members of the Computer and Communication Research Laboratory for all their help and discussions. I am also grateful to the sponsors who helped support my research; the National Science Foundation, Ascom Timeplex, Bellcore, BNR, DEC, Goldstar, Information & Communications, Italtel SIT, NEC, NTT, Synoptics Communications and Tektronix. Finally I would like to thank my wife, Chrysanthe Preza, for reading early drafts of the thesis and for her love and support through all this hard work.

10. BIBLIOGRAPHY

[1] Hamid Ahmadi, and Wolfgang E. Denzel. “A Survey of Modern High-Performance Switching Techniques.” *IEEE Journal on Selected Areas in Communications,* 7(7):1091-1103, September 1989.

[2] M. Ajmone Marsan, Gianfranco Balbo, and Giorgio Bruno. “TOPNET: A Tool for the Visual Simulation of Communication Networks,” *IEEE Journal on Selected Areas in Communications*, 8(9):1735-1747, December 1990.

[3] M. Atiquzzaman, and M.S. Akhtar, “Performance of Buffered Multistage Interconnection Networks in Non-Uniform Traffic Environment.”, *7th International Parallel Processing Symposium*, Newport Beach, California, April 1993, pages 762-767.

[4] Kevin Batcher. “Sorting Networks and Their Applications,” P*roceedings of the Spring Joint Computer Conference*, 1968, pages 307-314.

[5] Peter C. Bell, and Robert M. O’Keefe. “Visual Interactive Simulation - History, recent developments, and major issues.” *SIMULAITON*, 49(3):109-116, September 1987.

[6] V. E. Benes. “Mathematical Theory of Connecting Networks and Telephone Traffic, “ *Academic Press*, New York, 1965.

[7] Kadaba Bharath-Kumar, and Parviz Kermani. “Performance Evaluation Tool(PET): An Analysis Tool for Computer Communication Networks,” In *IEEE Journal on Selected Areas in Communications*, 2(1):220-226, January 1984.

[8] Giuseppe Bianchi, and Jonathan S. Turner. “Improved Queueing Analysis of Shared Buffer Switching Networks,” IEEE/ACM Transactions on Networking, August 1993, pages 482-490.

[9] B. Bingham, and H. Bussey. “Reservation-Based Contention Resolution Mechanism for Batcher-Banyan Packet Switches,” *Electronics Letters*, June 1988.

[10] Ian F. Blake. *An Introduction to Applied Probability*, John Wiley and sons, New York, 1979.

[11] P. Bratley, B.L. Fox, and L.E. Schrage. *A Guide to Simulation*, Prentice Hall, New Jersey, 1984.

[12] Jean-Yves Le Boudec. “The Asynchronous Transfer Mode: a tutorial.” Computer Networks and ISDN Systems.” 24:279-309, 1992.

[13] Richard G. Bubenik, and Jonathan S. Turner, “Performance of a Broadcast Packet Switch,” IEEE Transactions on Communications,” 37(1):60-69, January 1989.

[14] D. G Cantor. “On Non-Blocking Switching Networks, “ *Networks*, 1971, pages. 367-377.

[15] Robert S. Cahn, Pao-Chi Chang, Parviz Kermani, and Aaron Kershenbaum. “INTREPID: An Integrated Network Tool for Routing, Evaluation of Performance, and Interactive Design,” *IEEE Communication Magazine*, 28(7):40-47, July 1991.

[16] Erhan Cinlar. *Introduction to stochastic processes*, Prentice-Hall, 1975.

[17] Charles Clos. “A Study of Non-blocking Switching Networks,” *Bell System Technical Journal*, 32(3):406-424,. 1953.

[18] Comdisco Systems Inc. “Block Oriented Network Simulator (BONeS).” data sheet, 1989.

[19] C. Day, J.N. Giacopelli, and J. Hickey. “Applications of Self-Routing Switches to LATA Fiber Optic Networks,” *Proceedings of the International Switching Symposium*, 1987.

[20] M. Decina, P. Giacomazzi, and A. Pattavina. “Shuffle Interconnection Networks with Deflection Routing for ATM Switching: The Open-Loop Shuffleout.” *Proceedings of 13th International Teletraffic Congress*, June 1991.

[21] Thomas A. DeFanti, and Maxine D. Brown. “Visualization: Expanding Scientific and Engineering Research Opportunities.” In *IEEE Computer*, 22(8):12-25, August 1989.

[22] D.M Dias, and J.R. Jump, “Analysis and Simulation of buffered Delta Network,” IEEE Transaction on Computer, 30(4):273-282, April 1981.

[23] Alexander Dupuy, Jed Schwartz, Yechiam Yemini, and David Bacon. “NEST: A Network Simulation and Prototyping Testbed,” *Communications of the ACM*, 33(10):63-74, October 1990.

[24] G.S.Fishman. *Concepts and Methods in Discrete Event Digital Simulation*, John Wiley and sons, New York, 1973.

[25] Cynthia A. Funka-Lea, Tasos D. Kontogiorgos, Robert J.T. Morris, and Larry D. Rubin. “Interactive Visual Modeling for Performance,” *IEEE Software*, 8(9):58-68, September 1991.

[26] Udai Garg, and Yo-Ping Huang, “Decomposing Networks for Performance Analysis,” IEEE Transactions on Computers, 37(3):371-376, March 1988.

[27] James N. Giacopelli, Jason J. Hickey, William S. Marcus, W. David Sincoskie, and Morgan Littlewood. “Sunshine: A High-Performance Self-Routing Broadband packet Switch Architecture.” *IEEE Journal on Selected Areas in Communications*, 9(8), October 1991.

[28] Stefano Gianatti, and Achille Pattavina, “Performance Analysis of Shared-buffered Banyan Networks under Arbitrary Traffic Patterns,” *Proceedings of IEEE Infocom’93*, San Francisco, March 1993, pages. 943-952.

[29] L. Goke, and G. Lipovski. “Banyan Networks for Partitioning Multiprocessor Systems, “*Proceedings of International Symposium on Computer Architecture*, 1973, pages 21-28

[30] Alan Huang, and Scott Knauer. “Starlite: a Wideband Digital Switch,” Proceedings of Globecom’84, December 1984, pages 121-125.

[31] Joseph Y. Hui. *Switching and Traffic Theory for Integrated Broadband Networks*, Kluwer Academic Publishers, 1990.

[32] Yih-Chyun Jenq. “Performance Analysis of a Packet Switch Based on a Single-Buffered Banyan Network.”*IEEE Journal on Selected Areas in Communications*, 1(6):1014-1021, December 1983.

[33] Hyong Sok Kim, and Alberto Leon-Garcia. “Performance of Buffered Banyan Networks Under Nonuniform Traffic Patterns.” In *IEEE Transactions on Communications*, 38(5):648-658, May 1990.

[34] B.W. Kleinrock. *Queueing Systems*, volume 1,John Wiley and sons, 1975.

[35] Hisashi Kobayashi. *Modeling and Analysis: An Introduction to system Performance Evaluation Methodology*, Addison-Wesley, 1981.

[36] D. H. Lawrie, “Access and Alignment of Data in an Array Processor,” IEEE Transaction on Computers, 24(12):1145-1155, December 1975.

[37] T.H. Lee. "Analytical models for performance evaluation of single buffered banyan networks under nonuniform traffic." In *IEEE Proceedings*, Part E, Computers and Digital Techniques, 138(1):41-47, January 1991.

[38] T.T Lee, "A modular architecture for very large packet switches," *IEEE Transaction on Communications*, 38(7):1097-1106, July 1990.

[39] Soung C. Liew. "Performance of Various Input-buffered and Output-buffered ATM Switch Design Principles under Bursty Traffic: Simulation Study," *IEEE Transactions on Communications*, 42(3):1371-1379, March 1994

[40] Dimitris Logothetis and K. Trivedi. "Transient Analysis of the Leaky Bucket Rate Control Scheme Under Poisson and ON-OFF Sources," *Proceedings of IEEE Infocom'94*, Toronto, June 1994, pages 490-497.

[41] B.Melamed, and Robert J.T. Morris. "Visual Simulation: The Performance Analysis Workstation," *IEEE Computer*, 18(8):87-94, August 1985.

[42] Riccardo Melen, and Jonathan S. Turner. "Nonblocking Multirate Networks," *SIAM Journal on Computing*, April 1989, pages 301-313.

[43] Steven E. Minzer. "Broadband ISDN and Asynchronous Transfer Mode (ATM)." *IEEE Communications Magazine*, 27(9):17-24, September 1989.

[44] John K. Ousterhout. "Tcl: An Embeddable Command Language," *Proceedings of the Winter USENIX Conference*, 1990

[45] Athanasios Papoulis. *Probability, Random Variables, and Stochastic Processes*, McGraw-Hill, 1984.

[46] J.H. Patel. "Performance of Processor-Memory Interconnection for Multiprocessor." In *IEEE Transactions on Computer*s, 30(10):771-780, October 1981.

[47] Achille Pattavina. "Nonblocking Architectures for ATM Switching.", *IEEE Communications Magazine*, February 1993. pages 38-48.

[48] Achille Pattavina, and Alberto Monterosso. "Performance Analysis of Multistage Interconnection Networks with Shared-Buffered Switching Elements for ATM Switching." In *IEEE Infocom '92*, May 1992.

[49] Martin de Prycker, "Asynchronous Transfer Mode: Solution for Broadband ISDN." *Ellis Horwood Limited*, New York, 1993.

[50] Charles H. Sauer, Edward A. MacNair, and James F. Kurose. "Queueing Network Simulation of Computer Communication," In *IEEE Journal on Selected Areas in Communications*, 2(1):203-220, January 1984.

[51] H.S. Stone. "Parallel Processing with the Perfect Shuffle." *IEEE Transactions on Computers*, February 1971, pages 153-161.

[52] Bjarne Stroustrup. *The C++ Programming Language*, Addison-Wesley, 1991.

[53] Hiroshi Suzuki, Hiroshi Hagono, Toshio Suzuki, Takao Takeuchi, and Susumu Ivasaki. "Output-buffer Switch Architecture for Asynchronous Transfer Mode." In *ICC'89, IEEE International Conference on Communications*, 1:99-103, June 1989.

[54] Ted Szymanski, and Salman Shaikh. "Markov Chain Analysis of Packet-Switched Banyans with Arbitrary Switch Sizes, Queue Sizes, Link Multiplicities and Speedups." In *IEEE Infocom '88: Proceedings of the Seventh Annual Joint Conference of the IEEE Computer and Communications Societies*, April 1988, pages 960-971

[55] Fouad A. Tobagi. ‘Fast Packet Switch Architectures for Broadband Integrated Services Digital Networks.” Proceedings of the IEEE, 78(1):133-167, January 1990.

[56] Fouad A. Tobagi, Timothy Kwok, and Fabio M. Chiussi. “Architecture, Performance, and Implementation of the Tandem Banyan Fast Packet Switch.”, *IEEE Journal on Selected Areas in Communications*, 9(8):1173-1193, June 1991.

[57] George E. Daddis Jr., and H.C. Torng. “A Taxonomy of Broadband Integrated Switching Architectures.” *IEEE Communications Magazine*, 27(5):32-42, May 1989.

[58] Jonathan S. Turner, and L.F. Wyatt, “A packet network architecture for integrated services,” Proceedings of GLOBECOM’85, San Diego, CA, November 1983, pages 2.1.1-2.1.6

[59] Jonathan S. Turner. “Fluid Flow Loading Analysis of Packet Switching Networks,” *Proceedings of the International Teletraffic Congress*, June 1988.

[60] Jonathan S. Turner. “Queueing Analysis of Buffered Switching Networks.” In *Proceedings of 13th International Teletraffic Congress*, June 1991.

[61] Jonathan S. Turner. “Design of a Broadcast Packet Switching Network.” In *IEEE Transactions on Communications*, 36(6):734-743, June 1988.

[62] Jonathan S. Turner. “A Proposed Bandwidth Management and Congestion Control Scheme for Multicast ATM Networks.” In *Computer and Communication Research Center Technical Report WUCCRC*-91-1, Washington University, St. Louis. May 1991

[63] Jonathan S. Turner. “Reseqencing Cells In an ATM Switch,” *Washington University Computer Science Department*, WUCS-91-21, 1991.

[64] Jonathan S. Turner. “Managing Bandwidth in ATM Networks with Bursty Traffic,” *IEEE Network*, September 1992, pages.50-58.

[65] Jonathan S. Turner. “An Optimal Nonblocking Multicast Virtual Circuit Switch,” *Washington University Computer Science Department*, WUCS-93-30, 1993.

[66] Jonathan S. Turner. “Design and Analysis of Switching Systems,” *Washington University Computer Science Department*, December 1993.

[67] Einir Valdimarsson, “Blocking in Multirate Interconnection Networks,” *IEEE Transactions on Communications*, 42(4):2028-2035, April 1994.

[68] John M. Vlissides. “Generalized Graphical Object Editing.” *Stanford University Technical Report CSL-TR-90-427*, June 1990.

[69] L.T. Wu, “Mixing Traffic in a Buffered Banyan Network.”, *Proceedings of 9th Data Communication Symposium*, Whistler Mountain, British Columbia, September 1985.

[70] Ellen Witte Zegura. “Architectures for ATM Switching Systems.”, *IEEE Communications Magazine*, February 1993. pages 28-37.

[71] Lynette Van Zijl, Deon Mitton, and Simon Crosby. “A Tool for Graphical Network Modeling and Analysis” *IEEE Software*, 9(1):47-54, January 1992.

[72] Y. S. Yeh, M. G. Hluchyj, and A.S Acompora. “The Knockout Switch: A Simple, Modular Architecture for HIgh-Performance Packet-Switching.” *IEEE Journal on Selected Areas in Communications*, 5:1274-1283, September 1987.

11. VITA

Biographical items on the author of the thesis, Einir Valdimarsson.

- Born March 26, 1963 in Reykjavik, Iceland.
- Attended University of Iceland from September 1983 to June 1984.
- Attended Washington University in St. Louis from August 1984 to present. Received the degrees Bachelor of Science in Computer Science and Bachelor of Science in Electrical Engineering in May 1987. Received the degrees Master of Science in Computer Science and Master of Science in Electrical Engineering in May 1990. Worked as a research assistant in the Computer and Communication Research Center from August 1987 to present.

December, 1994

Short Title: ATM Switching System Performance , Valdimarsson, D.Sc. 1994