Order Number 9119655

Performance analysis of integrated broadband packet switching

Mathew, Johny, Ph.D.

City University of New York, 1991

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

PERFORMANCE ANALYSIS OF INTEGRATED BROADBAND PACKET SWITCHING

by

JOHNY MATHEW

A dissertation submitted to the Graduate Faculty of Engineering in partial fulfillment of the requirements for the degree of Doctor of Philosophy, The City University of New York.

1991

This manuscript has been read and accepted for the Graduate Faculty in Engineering in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

11/14/1990

Date

Professor T. N. Saadawi

Chairman of Examining Committee

11/14/90

Date

Dean G. Lowen

Executive Officer

Professor J. Barba

Professor M. Conner

Professor M. Lee

Professor D. L. Schilling

Dr. B. Kraimeche

Supervisory Committee

The City University of New York

Abstract

PERFORMANCE ANALYSIS OF INTEGRATED BROADBAND PACKET SWITCHING

by

Johny Mathew

Advisor: Professor Tarek N. Saadawi

Two broadband packet switches have been proposed and analyzed. A packet switch based on the idea of assignment on demand has been proposed and analyzed in Part I. The switch model assumes that switch connection can be reconfigured by a control unit according to a switching schedule. The switching schedule may either be provided by an algorithm which will optimize performance or can be a sequence of fixed switching configurations. In both cases, queueing analysis has been carried out. In the analysis, we have assumed that buffer capacity is limited and that the transmission period (frame length) is variable. The analysis is based on a discrete time Markov chain from which state probabilities of the transmission period have been derived. Also, buffer overflow probabilities have been obtained. Delay versus traffic characteristics with different buffer sizes have been presented and compared for each of the above cases. The main advantage of this switch model in comparison with other architecture is that in our model the switch connections are software driven and thus can optimize one parameter or another of switch performance.

In Part II an N x N nonblocking time multiplex switch handling both circuit switched and packet switched traffic is proposed and analyzed. We consider a system in which the incoming circuit traffic that can not be routed within a frame size is blocked and the incoming packet traffic which can not be served immediately is queued at the end of the existing input queue. Circuit switched traffic performance of the proposed switch is analyzed in terms of call blocking probability. Packet switched traffic performance is analyzed in terms of mean packet delay. A two dimensional Markov chain is used to model the system. It has been shown, through a special case, that the mean packet queueing delay can be reduced considerably by using movable boundary strategy as compared to fixed boundary strategy. A comparison between the performance of integrated switching strategy and circuit switching strategy has also been provided. It is shown that the circuit blocking probability has improved when integrated with packet traffic.

ACKNOWLEDGEMENTS

I wish to express my deepest gratitude to my mentor, Professor Tarek Saadawi for his interest, encouragement and guidance during the progress of this research. The many technical discussions and his constructive suggestions are most appreciated.

I would like to thank all the members of my doctoral committee; Dean Gerald Lowen, Professor Joseph Barba, Professor Michael Conner, Professor M. Lee, Professor Donald Schilling and Dr. Belka Kraimeche for their guidance and support. Special thanks to Dr. Gerakoulis for discussing and sharing ideas throughout this work. Thanks to Professor Paul Karmel for his guidance in my early years of graduate study.

Many thanks to my family, friends and colleagues especially those who are very close to me for their love and encouragement. Special thanks to my parents whose sacrifice and motivation led me to higher education, and to them I dedicate this work.

Table of Contents

PART II

List of Figures

1 INTRODUCTION

Traditionally, voice, data and video communications have been handled by separate specialized networks. The main reason for these specialized networks are the different characteristics of voice, data and video signals. Voice is real time analog signals generated by human speakers. While voice traffic can tolerate a certain amount of degradation and occasional blocking it can not tolerate large and variable transmission delays. Data traffic is mainly computer generated and are digital. Usually it can tolerate substantial delays but not errors. Video is like voice with much higher bandwidth requirement. By offering all these services in an integrated network provide convenience, flexibility, and better resource utilization. The advantages of an integrated communication system which can accommodate a variety of diverse services with different bandwidth requirements has been recognized for some time.

High performance packet switching has been a subject of intense research interest as well as practical importance, largely because of the increasing demand for computer communication. Space-division packet switching is emerging as a key component in the trend toward high-performance integrated communication networks for data, voice, image, and video [1], [2] and multiprocessor interconnects for building highly parallel computer systems [3], [4]. Unlike present-day packet switch architectures with throughputs measured in 1's or at most 10's of Mbits/s, a space-division packet switch can have throughputs measured in 1's, 10's, or even 100's of Gbits/s. These capacities are attained through the use of a highly parallel switch fabric coupled with simple per packet processing distributed among many high-speed VLSI circuits.

Conceptually, a space-division packet switch is a box with N inputs and N outputs that routes the packets arriving on its inputs to the appropriate outputs. At any given time, internal switch points can be set to establish certain paths from inputs to outputs; the routing information used to establish input-output path is often contained in the header of each arriving packet. Packets may have to be buffered within the switch until appropriate connections are available; the location of the buffers and the amount of buffering required depend on the switch architecture and the statistics of the offered traffic.

Clearly, congestion can occur if the switch is a blocking network, that is, if there are not enough switch points to provide simultaneous, independent paths between arbitrary pairs of inputs and outputs. A Banyan switch [4],[5], for example, is a blocking network. In a Banyan switch, even when every input is assigned to a different output, as many as $\sqrt{N}$ connections may be contending for use of the same center link. The use of a blocking network as a packet switch is feasible only under light loads or, alternatively, if it is possible to run the switch substantially faster than the input and output trunks.

A simple example of a nonblocking switch fabric is the crossbar interconnect with N^2 switch points. Here it is always possible to establish a connection between any idle input-output pair. Even with a nonblocking interconnect, some queueing in a packet switch is unavoidable, simply because the switch acts as a statistical multiplexor; that is, packet arrivals to the switch are unscheduled. If more than one packet arrives for the same output at a given time, queueing is required. Depending on the speed of the switch fabric and its particular architecture, there may be a choice as to where the queueing is done: for example, on the input trunk, on the output trunk, or at an internal node.

Fast packet is a connection oriented packet switching mechanism which achieves high throughput and low delay by reducing the processing required per packet to an absolute minimum [1] and then implementing it in hardware. Routing is performed at call setup and a virtual circuit is allocated which is fixed for the duration of the call. All flow control and error recovery protocol functions are performed on an end-to-end basis. The packet length across any virtual circuit is constant and small, and the packet format is very simple: a packet header containing a priority field and a label, (to identify the virtual circuit), of fixed length, e.g., 16 bits, followed by the information component typically in the region 4-64 octets.

Two fundamental components are required to construct a fast packet switch: switching and buffering. This results in three possible classes of packet switch design: input buffered, in which the buffering precedes the switching using a nonbuffered switch fabric [6], [7]; output buffered, in which the buffering follows the switching also using a nonbuffered switch fabric [8]-[10]; and the buffered switch fabric where buffering occurs internally within the switch fabric [11], [12]. Pure input buffering has a performance which is approximately 58 percent that of pure output buffering [13]-[15], for uniform traffic. However, pure output buffering requires an order of magnitude, more hardware, and switch fabric interconnections than does an input buffered solution [8].

Before we examine the fast packet switches in detail let us briefly go through the other existing switching technologies. Figure 1.1 displays a spectrum of techniques available for the transport of integrated services [2], [16]. In general, techniques towards the right end of this spectrum provide increasing flexibility to handle variable rate and bursty information, while requiring more processing. An area in the center of this spectrum is designated as

"statistical switching", representing techniques that provide transport of bursty and variable rate information without the full functionality of conventional packet switching. Each technique is now discussed in more detail.

1.1 CIRCUIT SWITCHING

Circuit switching dominates today's telecommunication facilities. In this technique, an entire circuit, as in analog communication, or a predetermined portion of a channel, as in time-division multiplexed digital communication, is dedicated to continuous use of the voice or data subscribers for the duration of the call. The public telephone network is an example of Traditional Circuit Switching. A person or a terminal places a call by entering into the switch the directory number (address) of the person or terminal to be called. The switch then sets up, via signaling messages, a dedicated connection between subscribers, consisting of a sequence of point-to-point circuits, joined together by switches at the junctions between them.

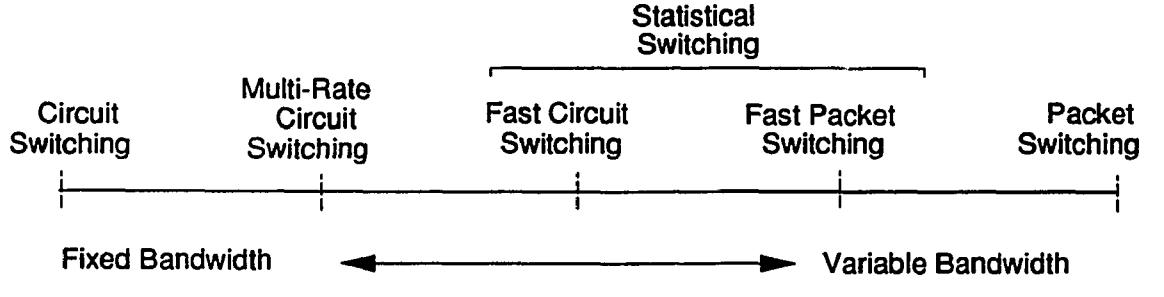


Fig. 1.1. Switching Technology Spectrum

In conventional circuit switched networks, the principal source of delay is the finite propagation speed of signals. This delay is very short for local calls but may be as much as 50 ms for a cross-country call, 100 ms for a terrestrial call between the U.S. and Europe or 250 ms for a call placed over a satellite. In addition, conventional switching and multiplexing equipment introduces a small fixed delay; this rarely exceeds a few milliseconds on an end-to-end basis.

1.2 PACKET SWITCHING

Conventional Packet Switching (PS) has been used in research and commercial data networks over the past 20 years providing error free data communication, with many additional features such as rate adaptation, protocol conversion, and tolerance of wide activity variations. These networks typically use a more complex internal structure than FPS. Errors in packet transmissions are corrected by retransmission on each link. Sophisticated protocols route packets around congested links and provide flow control between customers with varying speed access. Error and flow control techniques provide high quality data service, but substantially increase the processing required and may introduce substantial delay for some packets. These factors have made it difficult to apply conventional packet switching techniques to voice, which requires low delay for all packets and can require very large switches.

The requirements of high quality packet data customers can be met though a combination of a simple internal network structure (such as provided by Fast Packet Switching or Fast Circuit Switching) or any of the other techniques, in combination with interfaces at

the edge of the network which perform the needed error and flow control. However, because retransmission takes place across the network rather than across a single link, additional delay and overhead can be encountered in comparison to the conventional technique using per link error control. The speed of packet links and protocols used affect the relative performance of link-by-link and edge-to-edge error correction.

1.3 MULTI-RATE CIRCUIT SWITCHING

To those with a background in telephony, Multi-Rate Circuit Switching (MRCS) seems a natural choice for new communications systems [16]. MRCS provides connections having bandwidths equal to an integer multiple of some basic rate, such as 8 or 64 Kb/s. The user specifies a transmission speed when the call is setup and the network provides enough channels to satisfy the request. As Kulzer and Montgomery [2] point out, there are several problems with MRCS. First is the choice of the basic rate; many services require a low rate such as 1 Kb/s, but this can imply a long delay due to the large frame size required for time-division multiplexing. It also implies a large overhead for establishing high speed connections, since these must be implemented using multiple channels, each of which must be set-up individually. Even with a fairly large basic rate of 1Mb/s, a video connection might require the establishment of 100 channels. MRCS is also illsuited to applications with bursty transmission characteristics. Applications such as remote file access require occasional transfer of bursts of data at high rates such as 10 Mb/s. Dedicating a high speed connection to such applications is costly and inefficient. Using a lower speed channel yields efficiency, but only by sacrificing performance.

1.4 FAST CIRCUIT SWITCHING

Fast Circuit Switching (FCS) extends the concept of circuit switching by allowing the network to respond to bursty information. With FCS, a circuit is not reserved when the customer makes a call, but is set up only when bandwidth is needed and torn down when there is no information to send, freeing the resources for other calls. Circuit setup and teardown can take place by using a fast signaling mechanism, such as Common Channel Inter-office Signaling (CCIS), to completely set up the call from scratch each time a circuit must be allocated. Alternatively, the set of switches along the route that a call follows can remember the call, noting the bandwidth required and the destination, when a call is set up. Such a "virtual circuit" is activated when the customer sends information into the network by replacing a header identifying the call in an idle channel towards the first switch. Each switch in the path of the call notes the allocation of an incoming channel to the call, allocates an idle channel towards the next switch to the call, connects the incoming channel to the newly allocated channel, and sends a header to the next switch. This approach has also been called "burst switching". [17]-[19]. In either case, if setup is fast enough, an end-to-end circuit is established with little delay.

With FCS a switch may not always be able to satisfy a request to activate a virtual circuit, because bandwidth is not reserved for it. A request that cannot be satisfied must be queued resulting in delay due to buffering or information loss.

Clearly, one can combine the concepts of MRCS and FCS, providing dynamic setup and teardown of multiple low rate channels. There is an attractive synergy to this arrangement.

FCS gives an alternative way of providing very low rate channels, thus eliminating the need for a small basic rate.

The combination of multiple rates and FCS can meet a wide range of needs. It is quite likely, however, that such a network would be very complex to design and maintain. Further, it would require switching and transmission facilities quite different from those currently in use. Also it does not allow the connection of customers operating at different rates, which is a common need in data communications.

1.5 FAST PACKET SWITCHING

Fast Packet Switching (FPS) is the next option on the spectrum. As in conventional packet switching, FPS uses the transmission facility as a "digital pipe," which carries short packets of information one after another. Information in the header of each packet identifies which of many logical connections the packet belongs to. With this multiplexing scheme, connections of arbitrary bandwidth are accommodated in a simple and natural way. A key aspect of FPS is the recognition that the high speed and low error rate of modern digital transmission facilities allow simplification of the communications protocols used in conventional packet switches. These simplifications make possible the construction of hardware protocol processors. High speed transmission facilities also dramatically reduce the queuing delays inherent in packet switching.

In the move toward high-performance packet switching for integrated service networks, attention is focusing on packet-switching architectures that provide many simultaneous

input/output paths through the switch fabric and allow the internal paths to be time-multiplexed in a statistical rather than deterministic fashion. Such architectures provide the capability for high-speed transmission (1-200 Mbits/s) on each input/output with a total switch capacity of 1-200 Gbits/s. Because of the high-speed operation of the switch, the processing of packets is largely hardware based, with packet headers containing address information that is used by the switch fabric to route packets from inputs to outputs on the switch. Depending on its design, the switch fabric may be blocking. That is the switch fabric may be unable to provide simultaneous, independent paths between arbitrary pairs of inputs and outputs. For example, see Figure 1.2. This form of blocking is known as internal link blocking. The internal link blocking refers to a case where packets are lost due to contention for a particular link inside the network. However, even if the switch fabric is nonblocking, congestion in the switch will still arise because, unlike a circuit switch, arrivals to a packet switch are unscheduled: two or more packets may arrive simultaneously on different inputs destined for the same output. This form of blocking is known as output port blocking. Figure 1.2 shows these effects in an example of an 8 x 8 delta 2 network. One of these contending packets for an output may be allowed to pass through the switch, but the others must be queued for later transmission on the output. This form of congestion is unavoidable in a packet switch and dealing with it often represents the greatest source of complexity in the switch architecture.

There are several ways to reduce the blocking or to increase the throughput of banyan-type switches:

increasing the internal link speeds relative to the external speeds,

placing buffers in every switching node,

using a handshaking mechanism between stages or a back-pressure mechanism to delay the transfer of blocked packets,
using multiple networks in parallel to provide multiple paths from any input to any output or multiple links for each switch connection [20], [21],
using a distribution network in front of the banyan network to distribute the load evenly.

In this section we examine different high performance packet switches. We try to classify them according to different approaches for providing statistical fluctuations in packet arrivals to the switch. We will classify them into the following categories:

*Input queueing.

*Output queueing.

*Queues at internal nodes.

*Completely shared buffering.

*Any combinations of the above.

In the following sections, we give a detailed descriptive overview of most switch architectures within each class.

1.5.1 SWITCH FABRICS WITH INPUT QUEUEING

As mentioned earlier, one drawback of the banyan networks is that they are internally blocking in the sense that two packets destined for two different outputs may collide in one of the intermediate nodes. However, if packets are first sorted based on their destination addresses and then routed through the banyan network, the internal blocking problem can be avoided completely. This is the basic idea behind the

sort-banyan type networks. Figure 1.3 shows the basic structure of a sort-banyan network. The first segment consists of a sorting network (in this case, a Batcher bitonic sort network [22]) which sorts the packets according to their destination address, followed by a shuffle exchange and a banyan network which routes the packets. It can be shown that packets will not block within the banyan network. Note that Figure 1.3 shows two connection patterns which would have caused internal link blocking in a pure banyan network (see Figure 1.2). Each node of the sort network as shown in Figure 1.3 is a 2 x 2 switching element which sorts the incoming packets in the order as indicated by the arrow shown. The number of elements in the sorting segment is $N/4((\log_2 N)^2 + \log_2 N)$. The network operates in a synchronous manner, and packets are processed in each stage in parallel. While this self-routing interconnection network is internally nonblocking, blocking still may occur if destination addresses are not distinct, i.e., if there are packets with identical addresses. Hence, simultaneous packets destined for the same output will collide within the banyan part of the network.

Recently, Hui [6], [7] proposed a switch architecture based on the sort-banyan network structure, but with a different scheme than Starlite (discussed later in sec 1.4.4) to overcome the output port contention problem. The advantage of this approach is that the switch fabric delivers exactly one packet to each output port from one of the input ports which request a packet delivery to the same output port. Hence, packets are never lost within the fabric and they are delivered in sequence.

The basic structure of this fabric, like the Starlite switch, is a Batcher sorting network

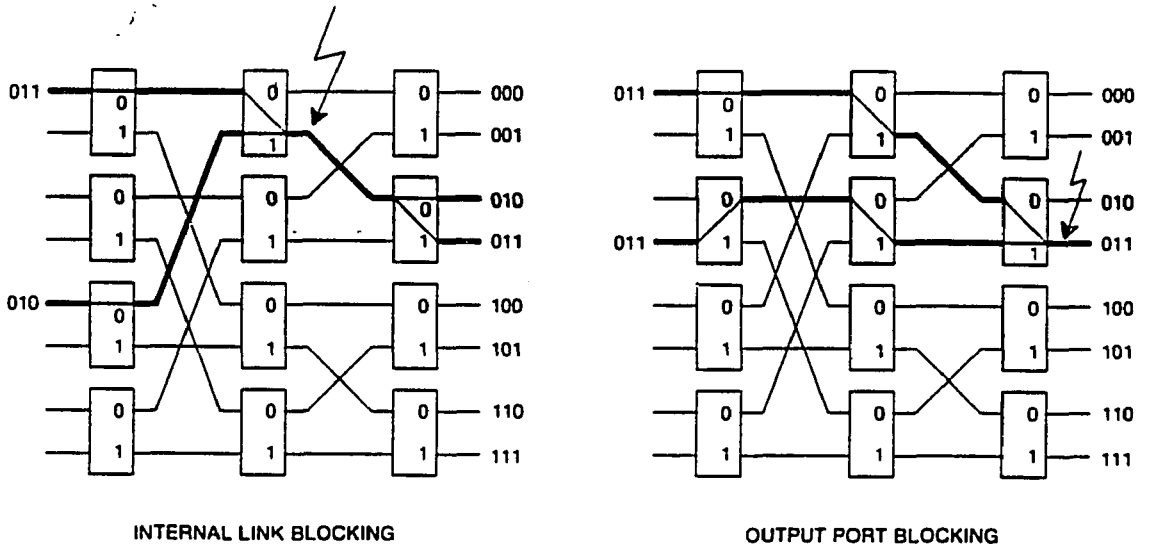


Fig. 1.2. Internal link blocking and output port blocking in an 8 × 8 delta-2 network.

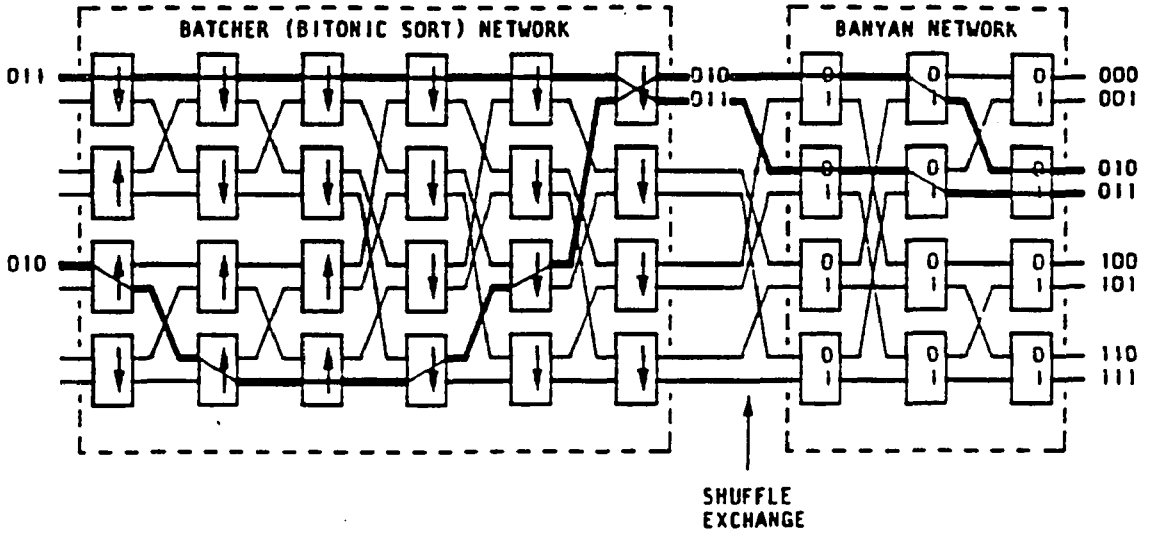
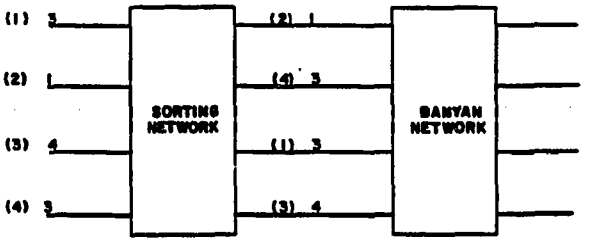


Fig. 1.3. Basic structure of a sort-banyan network.

followed by a banyan routing network. The combined sort banyan network is internally nonblocking. To resolve the output port conflict, a three-phase algorithm in conjunction with input queueing at each input port is employed. All packets are of a constant length and the switch operates in a synchronous manner. In the arbitration phase (Phase 1) of the algorithm, each input port i sends a short request packet, which is just a source-destination pair (i,j_i). (See Figure 1.4). The requests are sorted in nondecreasing order according to the destination address j_i, and the request is granted only if j_i is different from the one above it in the sorted list.

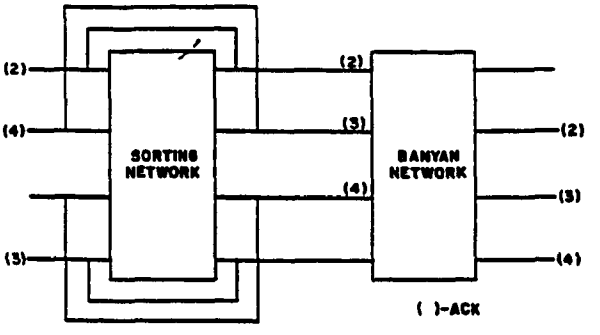
However, the input port which made the request does not know the result of the arbitration. Consequently, the request packet (i,j_i) which won the arbitration must send an acknowledgment packet to input port i via an interconnection network. This process constitutes the acknowledgment phase (Phase II). By bringing a fixed connection from the k^{Th} output of the Batcher network to the k^{Th} input of the Batcher network (Figure 1.5), the request packet (i,j_i) may send an acknowledgment packet from input port k, the position of the request packet in the sorted order, to input port i. The acknowledgment packets are send to distinct output ports since each input port can send at most 1 request in Phase I. Again the Batcher-banyan network is used to acknowledge the input ports.

Input ports receiving acknowledgments for their request then transmit the full packet in the final Phase III (Figure 1.6) through the same Batcher-Banyan network, without conflict at the output port. Input ports which fail to receive an acknowledgment retain the packet in a buffer for retry in the next slot, when the 3-phase cycle is repeated.



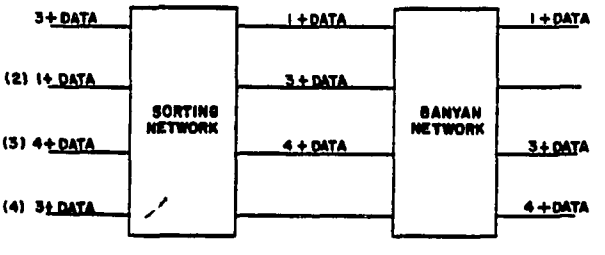
PHASE I: SEND AND RESOLVE REQUEST
- SEND SOURCE-DESTINATION PAIR THROUGH SORTING NETWORK
- SORT DESTINATION IN NON-DECREASING ORDER
- PURGE ADJACENT REQUESTS WITH SAME DESTINATION

Fig.1.4. Phase I of 3-phase algorithm.



PHASE II: ACKNOWLEDGE WINNING PORT
- SEND ACK WITH DESTINATION TO PORT WINNING CONTENTION
- ROUTE ACK THROUGH BATCHER-BANYAN NETWORK

Fig. 1.5. Phase II of 3-phase algorithm.



PHASE III: SEND PACKET
- ACKNOWLEDGED PORT SEND PACKET THROUGH BATCHER-BANYAN NETWORK

BUFFERS AT PORT CONTROLLER

Fig. 1.6. Phase III of 3-phase algorithm.

The queueing of the unacknowledged packet at the head of the queue will cause the subsequent packets which might have been intended for a free output port to be delayed. This phenomenon is called head of the line (HOL) blocking, illustrated in Figure 1.7. Notice that no request at the head of the line seeks delivery to output port 2, while a request for output port 2 is blocked at input port 1 because the HOL request at input port 1 is blocked. HOL blocking reduces throughput of the switch.

The switch supports a port speed of 150 Mbits/s. Of course, phases one and two constitute overhead processing for the switch fabric. Therefore, the switch fabric would have to be speeded up by a fraction which depends on the size of the switch fabric and the size of the information field of the packet. For example, as shown in [6], for a 1000 x 1000 switch and the packet size of 1000 bits, the overhead is about

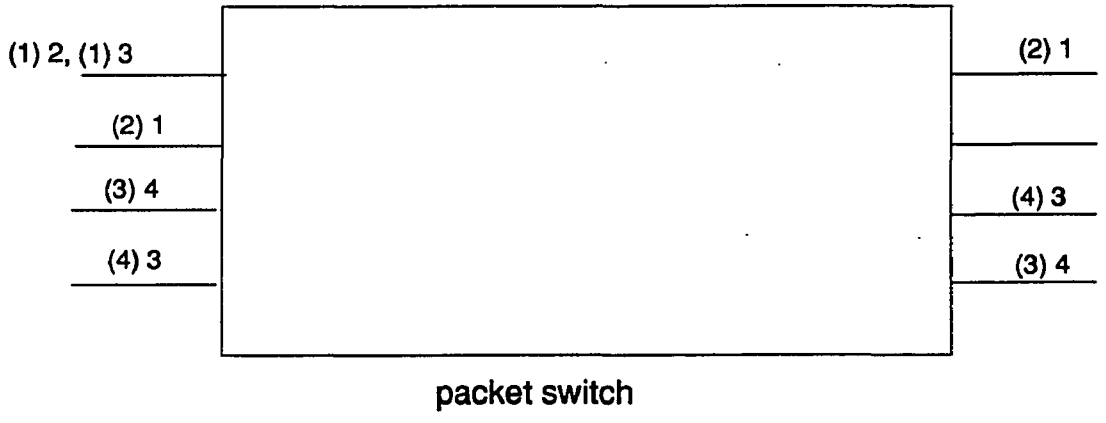


Fig. 1.7. Output conflict and head of line blocking

14 percent. This means that the switch has to operate at 170 Mbits/s in order to handle a 150 Mbit/s input port speed.

The performance study of this switch for random traffic indicates that the maximum throughput is about 58 percent. Actually, this is a theoretical limit which can be shown for any nonblocking space-division switch operating in a packet-switched mode which employs input queueing with random traffic (see [13]). Results in [6] also indicate that, with a reasonable input buffer size, an acceptable buffer overflow probability can be achieved.

The next switch under consideration uses a crossbar switch fabric. Crossbar switches have always been attractive to switch designers because they are internally non-blocking and they are simple. In addition to circuit-switch applications, they have also been considered as a base for switches which operate in a packet mode. But unfortunately, the simple crossbar matrix has the property of square growth and is not economical for large switches. Nearly all known approaches are either designed for relatively small applications or just for a building block which is used in larger multistage arrangements.

Even though the crossbar matrix is internally nonblocking, in packet mode operation, the probability of output port contention remains. Hence, a queueing function has to be added to the pure crosspoint matrix in order to overcome that problem. The location of this queueing function allows the approaches of crossbar-based switches to be

categorized. In principle, there are three possibilities, namely, crossbar matrices with input queueing, matrices with queueing within the crosspoints themselves, and matrices with output queueing.

If the queueing function is located at the inputs of the crossbar matrix, then a switch control is necessary which arbitrates all packets waiting at the heads of the different input queues and which are destined for a certain output port.

One solution is a centralized control. This control gets requests over separate control paths from all switch input ports which have packets waiting. It schedules these requests, sets up the necessary crosspoints in the matrix, and grants the requests as soon as the packets can be transmitted. This way, the matrix itself remains as simple as possible and can be realized economically with high-speed technology. By use of several matrices in parallel, even higher speeds can be achieved. Certainly, the central control is the bottleneck of this approach. The control overhead drastically increases with the size of the switch. Nevertheless, with suitable technology and intelligent techniques (e.g., pipelining), very high-speed, centrally controlled packet switches can be built, as long as they are small.

One way to reduce the control overhead is to distribute the control function [23], See Figure 1.8. There, each output port has its individual control, called arbiter. This arbiter allows only one of the input ports to be switched through at the same time to the corresponding output port by a fair algorithm. This is accomplished by a separate control signal (back pressure) to each input queue which practically stops all but one

of those queues which compete for the same output port. Since only one of the arbiters generates a back-pressure signals to a specific input queue at the same time, all of these signals of a corresponding matrix line can be interconnected very simply by a wired OR connection. In addition, this switch proposal does not need separate request

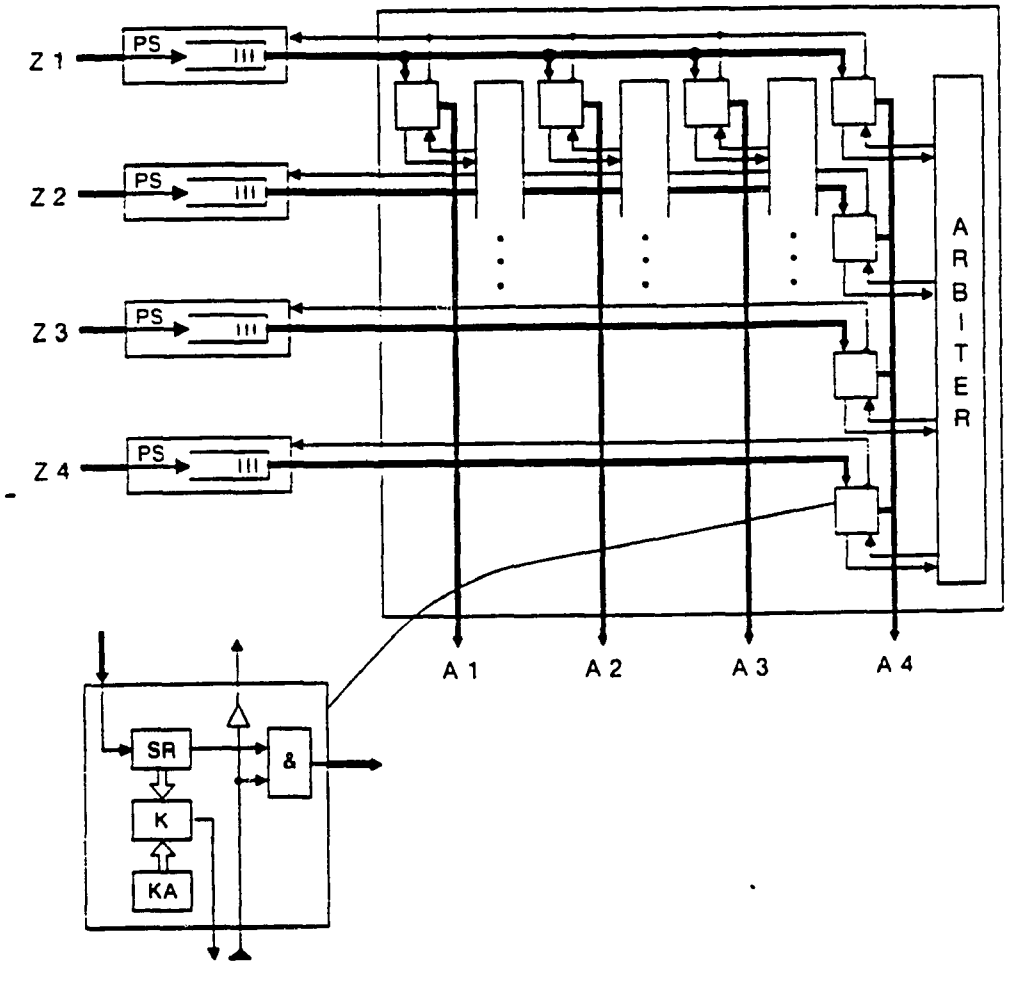


Fig. 1.8. The crossbar switch with input queueing

lines from the input queues to the arbiters because it has the address decoder function distributed in the crosspoints of the matrix. Hence, this approach represents a self-routing crossbar switch.

The performance of the crossbar matrices with input queueing depends very much on the control overhead, and especially on the way in which the input queues are organized. If the input queue is a simple, single FIFO queue, then the throughput of the switch saturates even at 58 percent (see [13], [14]) or less, depending on the control overhead and the traffic distribution. This is due to the head of the line (HOL) blocking phenomenon, already mentioned earlier. This means that the waiting packet at the head of the queue eventually blocks subsequent packets which might be destined for momentarily free output ports. But, if the input queue is organized as multiple queues, i.e., one per destination port, then we come closer to the concept of output queueing, even though these queues are physically located at the inputs. As mentioned earlier, output queueing provides ideal performance. The extent to which the performance of input queueing with multiple queues per input measures up to that of output queueing depends largely on how the input queues are controlled and scheduled and what the control overhead is.

1.5.2 SWITCHING FABRICS WITH OUTPUT QUEUEING

The switch fabrics described in this section are based on a fully interconnected topology in the sense that every input has a non overlapping direct path to every output so that no blocking or contention may occur internally. They employ output

queueing in order to resolve the output port contention. Intuitively speaking, in any nonblocking space-division packet switch, a higher throughput can be achieved with output queueing as compared to input queueing. This is because the head of the line blocking effect which is the limiting factor in a switch with input queues disappears with output queueing. This result has been shown analytically by Karol and Hluchyj in [13], [14]. In fact, one can show that a nonblocking space-division switch with an output queueing of infinite capacity would give the best delay/throughput performance.

The first switch fabric we describe in this class is the Knockout switch [8]-[10]. The Knockout switch is designed for a pure packet-switched environment and can handle either fixed-length (called Knockout I [8]) or variable-length packets (called Knockout II [9]). The Knockout switch uses one broadcast input bus from every input port to all output ports as shown in Figure 1.9. Each output port has a bus interface which can receive packets from each input bus line or input port. Hence, no contention occurs between packets destined for different outputs. In addition, simultaneous packets from several inputs can be transmitted to the same output. In Figure 1.9, one of the output bus interfaces is shown in more detail. It has three major components. The first component is the set of N packet filters, each interfacing a bus line. The packet filters, which implement the self-routing function, detect the address of each packet on the broadcast bus and pass those destined to that output on to the next component which is the concentrator. The concentrator uses a novel algorithm to select a fixed number of packets, say L, from the N incoming lines to the concentrator. The L selected packets are stored in the order of their arrivals into a shared buffer

which constitutes the third component. The main philosophy behind the N to L concentration mechanism is that the probability of packet loss due to output congestion can be kept below the loss expected from other sources such as channel errors. This means, for example, that with N large and L=8, a packet loss rate of less than 10^{-6} can be achieved. Taking advantage of this observation, the number of separate buffers needed to receive simultaneously arriving packets is reduced from N to L. This result holds under the assumption of uniform traffic patterns. If the traffic pattern is nonuniform, L has to he higher (e.g., up to 20) for the same packet loss rate, depending on the nonuniformity of traffic [10].

The basic principle of selecting L packets out of N possible contenders in the concentrator stage is an algorithm implemented in the hardware analogous to a knockout tournament. For a concentrator with N inputs and L outputs, there are L rounds of competition. The basic building block of the concentrator is a 2 x 2 contention switching element, with one output being the winner and one the loser. When two packets arrive, one is selected randomly as a winner. Figure 1.10 shows a block diagram of an 8:4 concentrator made of these 2 x 2 contention switches. The boxes marked "D" indicate a one bit delay line to keep the competition synchronous. The first round of tournament starts with N contenders. The N/2 winners from the first round advance to the second round. The winners in the second round advance to the third round, and so on. Note that the final winner at the output number has won all rounds, the winner at output number two has won all but one round, and so on.

The next segment after the concentrator is a shared buffer structure. The output buffer

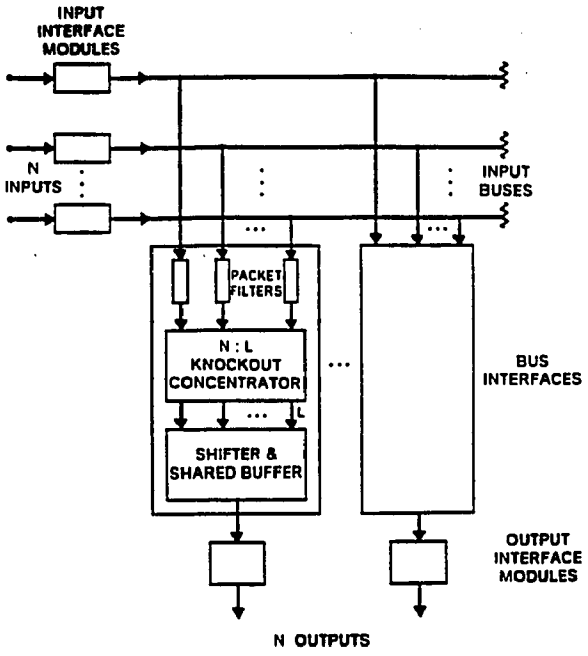


Fig. 1.9. Basic structure of the Knockout switch.

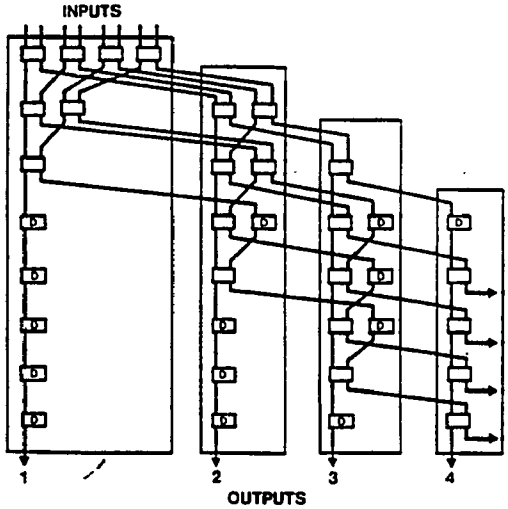


Fig. 1.10. Example for an 8:4 Knockout concentrator.

must be capable of storing up to L packets within one packet time slot. This is achieved by using L separate buffers preceded by a shifter function. This logically implements an L-input single-output FIFO queueing discipline for all packets arriving at the output.

The design of the Knockout switch is based on possible VLSI realization with input/output and internal hardware operating at 50 Mbits/s. Also, a solution is proposed for modular growth which can grow from 32 x 32 to 1024 x 1024.

Another switch fabric, the design of which is based on an interconnection structure with no internal blocking and queueing capability at the outputs of its modules, is the Integrated Switch Fabric proposed in [24]. This switch fabric is designed to handle both circuit-switched and packet-switched traffic in a unified manner. It is self-routing, and uses uniform fixed-length minipackets within the switching fabric for all types of connections. Circuit-switched connections can be provided for various speeds and for constant delay with full transparency to the terminal ports. Its design concept emphasizes modularity; it is based on VLSI technology, aiming for a single chip as the basic building block such that a very large range of switch fabrics can be configured, covering sizes from 16 to more than 1000 input/output ports. This translates into a throughput capability from 500 Mbits/s to 30 Gbits/s assuming a speed of 32 Mbits/s per port.

Figure 1.11 shows the basic structure of the fabric which consists of two major components; the switch fabric adapters (SFA) and the switch fabric elements (SFE).

The function of the switch fabric adapter is to convert the user information from packet-switching and circuit-switching interfaces into uniform fixed-length minipackets. Figure 1.11 also shows the basic structure of a k x k switch fabric element which consists of a self-routing segment of an output queueing segment. The self-routing segment of an SFE performs the minipacket routing function. It is a decoder with a tree structure per input which can simultaneously route minipackets from every input to every output. Each node of the tree decoder segment has one input and two outputs and works like a simple banyan node with only one used input. Therefore, within each module, up to k minipackets can be transferred to the same output at the same time. The interface between the terminating points of all trees belonging to a certain SFE output and the output line consists of k small shift registers (SR) for intermediate storage of minipackets and a pair of FIFO queues which constitute the output queueing segment. The importance of the shift registers is to store the minipackets momentarily to allow sequential access to the output FIFO queue pair. The contents of the shift registers are transferred sequentially into the associated output FIFO at a speed k times higher which is realized by parallelism. The combination of shift registers and their corresponding output FIFO is a realization of a multi-input single-output FIFO queue. One major difference between this switch fabric and most others is that it uses a priority scheme for different classes of traffic. Two priority classes are proposed. One is used for the high-priority or time-critical type traffic, and the other for the low-priority traffic. Circuit-switched connections are supported with minipackets which have high priority. Another feature of this switch architecture is its simple modular growth capability. That is, larger switch sizes can be configured by combining k x k switch fabric elements via stage-

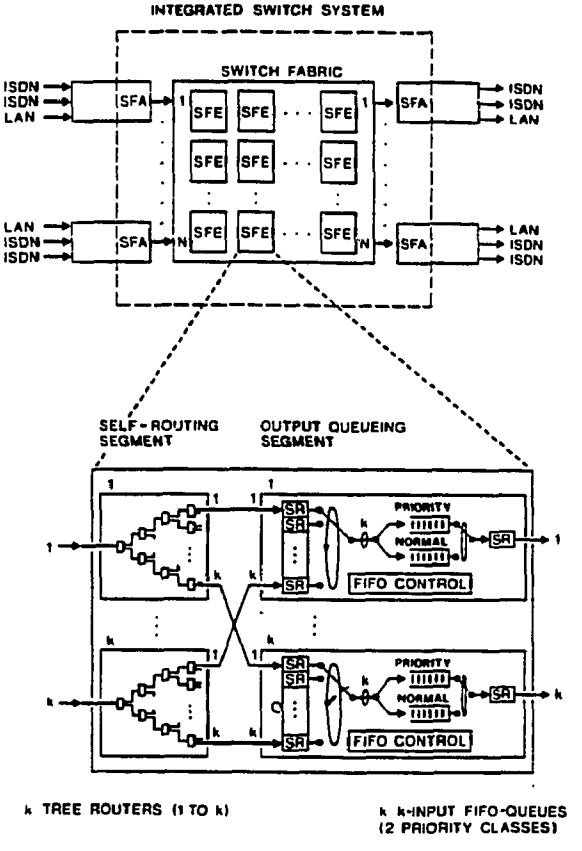


Fig. 1.11. Basic switch structure and switch fabric element propsed in [24]

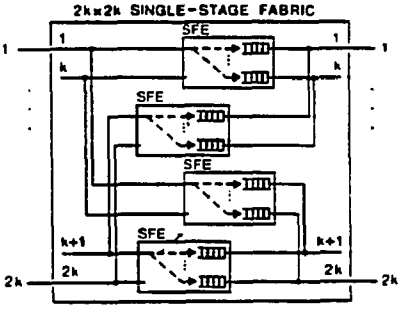


Fig. 1.12. Single-stage expansion in the switch proposed in [24]

expanding and/or multistage arrangements.

A single-stage N x N switch fabric can be built from the basic k x k element. Figure 1.12 shows an example of a 2k x 2k switch fabric configured from four k x k switch fabric elements. Each basic module has a selection logic which can be set that only the appropriate module accepts the incoming minipackets and routes them internally to the destined output. Furthermore, in each module, there are provisions for each output to ensure that only one output FIFO queue is feeding the output line at a time. In the same manner, a 4k x 4k fabric can be realized from a 2k x 2k fabric, and so on. In general, to build a single-stage N x N fabric from a basic k x k element, $(N/k)^2$ modules are required. It should be noted that a single-stage switch fabric which is realized this way still preserves its disjoint-path topology with output queueing. However, each output of the N x N single-stage configuration has one logical queueing segment which is physically realized in different switch fabric elements as parallel queues. For a moderate-size switch, say up to 128 x 128, the single-stage approach seems feasible and reasonable. But, for a large-size switch, a three-stage realization is proposed which yields a very cost-effective solution.

Finally let us look at crossbar switching matrices with output queueing. If the switch fabric runs N times as fast as the input and output trunks, all the packets that arrive during a particular input time slot can traverse the switch before the next input slot, but there will still be queueing at the outputs. Under this condition it is not attractive for very high speed switches.

1.5.3 SWITCH FABRICS WITH QUEUES AT INTERNAL NODES

Figure 1.13 Shows the concept of a crossbar switch which implements the queueing function within the crosspoints themselves. The simple on/off switch of a crosspoint is replaced now by a FIFO queue preceded by an address decoder function (packet filter). So, the packets can be sent directly into the matrix. A packet can only pass through the filter whose address matches the packet's destination address. So, the self-routing concept is realized. The queues of a matrix column which belong to one specific output port have to be emptied in a fair way, e.g., in round-robin fashion. This is also done by one very simple arbiter per output port

Figure 1.14 Shows the concept of the Bus Matrix Switch proposed in [11]. The PPD (primary packet distributor) interfaces incoming packet ports and the SPD (secondary packet distributor) interfaces outgoing packet ports. Each PPD and SPD has its own packet transmission bus. These buses are arranged in a matrix. At cross points of the incoming and outgoing buses, there is an XPM (cross-point memory) with packet queueing buffers.

The PPD receives a packet from the I-PPU, determines the destination XPM and outgoing SPD by referring packet transmission header, then sends it to the XPM. The SPD scans every XPM on its bus, removes packets from the XPM, and sends them to the O-PPU. Because XPM's are realized with dual port buffers, their inputs and outputs can operate independently. Thus, every PPD and SPD operates independently

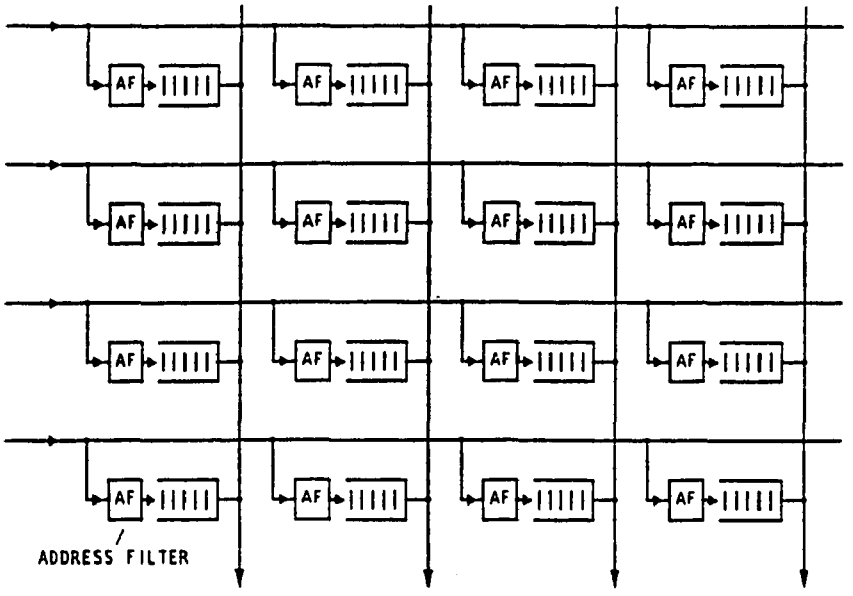


Fig. 1.13. A crossbar switch with queueing in the crosspoints.

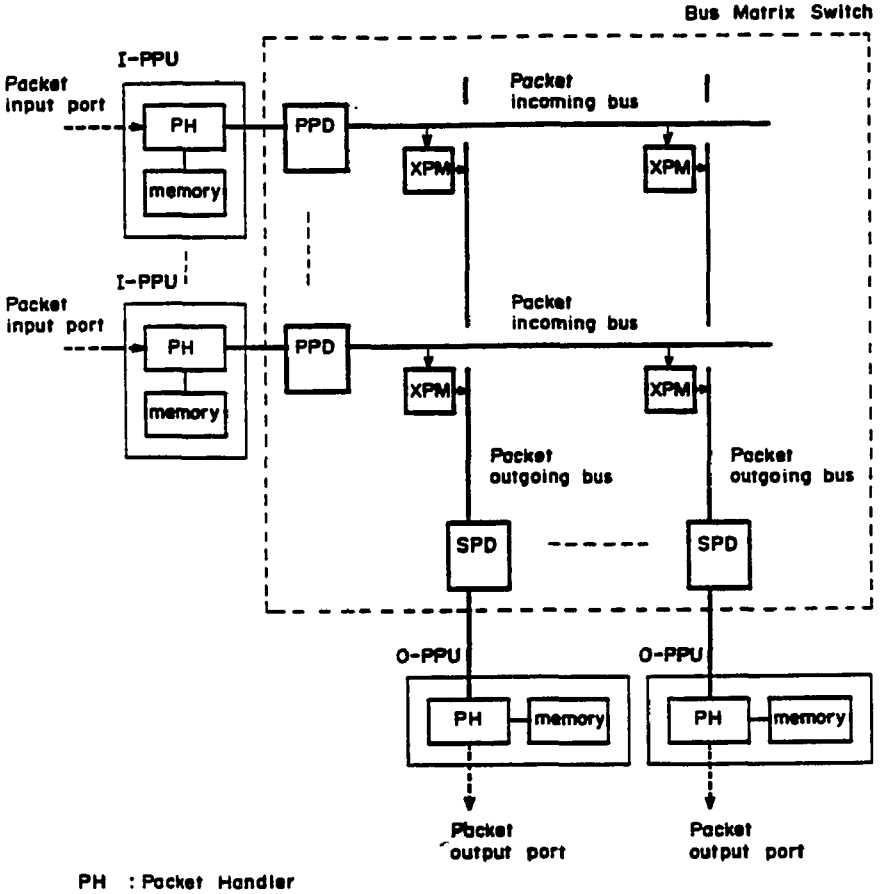


Fig. 1.14. Bus matrix switch.

and asynchronously. Thanks to this, the packet switching on each bus is realized in parallel. Also, the length of a packet can be variable.

The basic XPM configuration can be realized using an FIFO buffer. If priority control is necessary, XPM can use RIFO (random-in first-out), FIRO (first-in random-out), or multiple FIFO. The XPM capacity should be determined depending on the network traffic design, which will not be described here. The switching path of a packet is defined by determining the relation between the PPD and one of the SPD's. Therefore, the PPD should have a switching definition table which performs translation from a packet transmission header to a destination SPD address. The SPD address can specify a single SPD or a group of SPD's. This group addressing enables broadcast functions to be easily achieved.

The system can be expanded by increasing the matrix scale, but that is limited by the hardware. If the maximum speed of an accommodated packet port is low enough in comparison to the bus capacity, a single bus can accommodate several ports with a total bandwidth which is less than the bus capacity. Figure 1.15 shows this configuration (Type-1). If the port speed is near the bus capacity, a grading scheme, commonly applied to cross-bar switches, can also be applied to the BMX switch. Using an N x N BMX switch as an element, multistage grading is possible. Using the three-stage configuration shown in Figure 1.16, (Type-2) the speed of linking between the stages may be the same as the input/output speed. Type-1 configuration is suitable for a local switch which accommodates many low-speed subscriber loops. Type-2 is applicable to tandem or broadband subscriber switches.

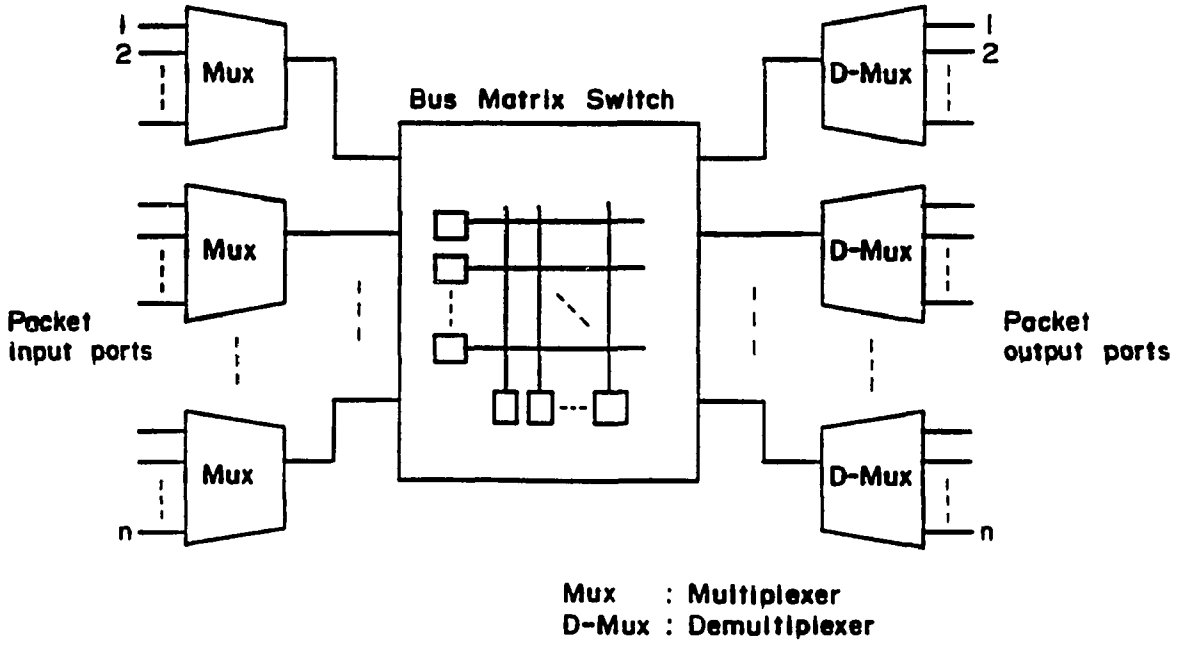


Fig. 1.15. Multiplexed-type configuration.

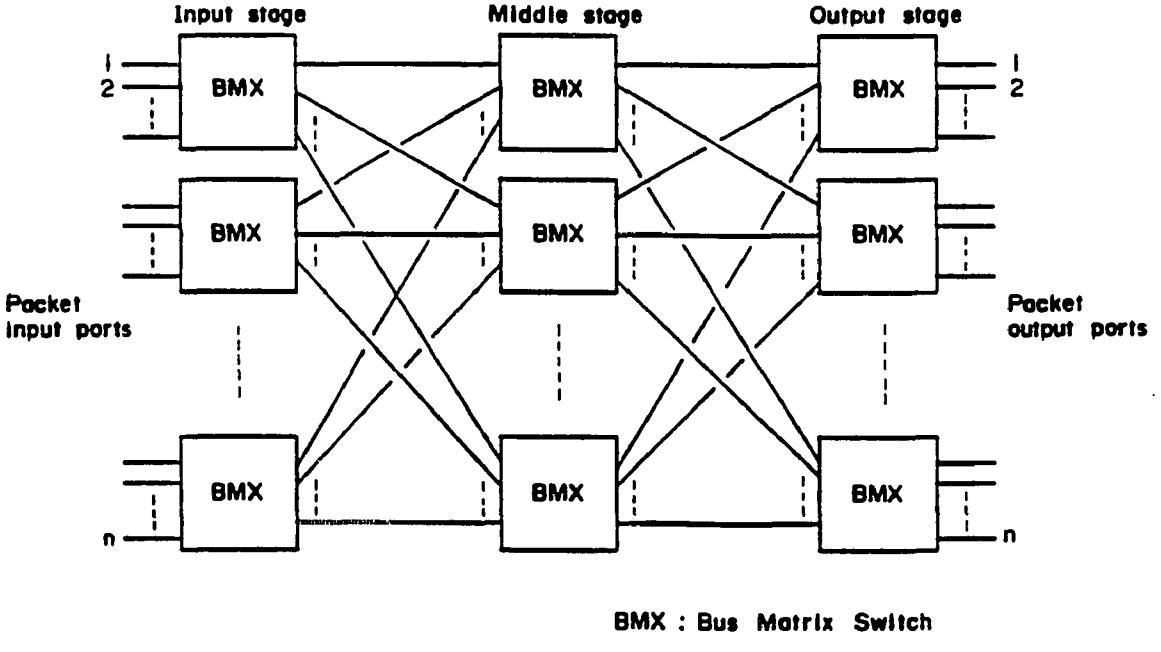


Fig. 1.16. Grading-type configuration.

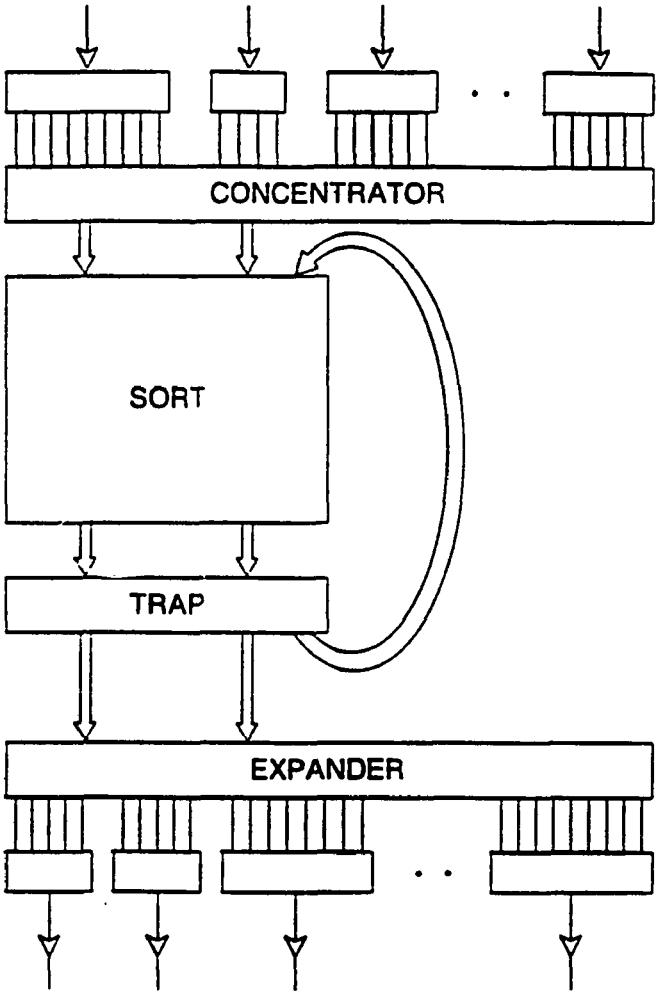


Fig. 1.17. Basic structure of the Starlite switch.

1.5.4 SWITCH FABRICS WITH COMPLETELY SHARED BUFFERING

The very first switch fabric implementation which proposed and employed the sort-banyan self-routing structure is the Starlite switch [25]; see Figure 1.17. At the switch interface, various services are converted into constant length "switch packets" with a packet header indicating the routing information which is the destination address. To overcome the output port contention problem, the Starlite approach uses a trap network between the sort and the banyan (called the expander here) segment which detects packets with equal destination addresses at the output of the sort segment. Thus, the packets with repeated addresses are separated from the ones with distinct addresses. The packets with repeated addresses fed back to the input side of the sort network for reentry within the next cycle. These packets can only use the idle input ports. Since the number of recycled packets at any time can be larger than the free input ports, a buffering stage must be used for the recycled packets if packet loss is not acceptable. The trap network consists of a single-stage comparator followed by a banyan network of the same type as the routing network. Note that if packets were simply recycled through the switch, they would be delivered out of sequence. This problem is solved by "aging" packets as they recirculate and by using the network to give old packets priority over new ones.

1.5.5 SWITCH FABRICS WITH INPUT AND OUTPUT QUEUES

Turner's Integrated Services Packet Network (ISPN) switch module is shown in Figure 1.18, [26]. The SM terminates up to 63 FOLs, engineered for a maximum occupancy of 80%, giving the SM a raw throughput of over 5 Gbits/s, or about 1,200,000 packets per second. The Packet Processors (PP) perform the link level protocol functions, including the determination of how each packet is routed. This includes routing connection control packets to the Connection Processor and datagram packets to one of several Datagram Routers (not shown). The Switching Fabric (SF) is the heart of the SM. It has a highly parallel organization and provides multiple paths for packets to ensure good performance. The SF also replicates broadcast packets as necessary and can support distribution of packets across a set of outgoing FOLs. The Connection Processor (CP), is responsible for establishing connections, including both point-to-point and broadcast connections. To do this, it exchanges control packets with CPs in neighboring SMs and controls the actions of the PPs and SF by writing information in their internal control tables. It also performs a variety of administrative and maintenance functions.

The structure of the Packet Processor is shown in Figure 1.19. It contains four packet buffers. The Receive Buffer (RCB) is used for packets arriving from the FOL and waiting to pass through the SF. The Transmit Buffer (XMB) is used for packets arriving from the SF that are to be sent out on the FOL. The Link Test Buffer (LTB) and Switch Test Buffer (STB) provide paths for test packets used to verify the

operation of the FOL and SF respectively. The RCB has a capacity of 16 packets, the XMB has a capacity of 32. The LTB and STB can each hold two packets. The four buffers require a total of about 240 Kbits of memory. The Logical Channel Translation Table (LCXT) is a dual port memory that implements a table with 4096 entries of four bytes each. It can be read by the Output Circuit and written by the Input Circuit. Logical channel translation is the process used to determine how to route a packet belonging to a connection through the SM. Each PP contains a Logical Channel Translation Table (LCXT) used for this purpose.

The Receive Circuit (RCV) converts the incoming optical signal in eight bit parallel format, synchronizes it to the local clock, discards packets with header errors, routes test packets to the LTB and other packets to the RCB. The Output Circuit (OUT) takes packets from the RCB, performs the logical channel translation described above and sends the packets on to the switch. It also processes test packets from the STB. The Input Circuit (IN) routes data packets to the XMB, removing the RF, SPP and C fields in the process. It also routes switch test packets to the STB, and updates the LCXT memory in response to LCXT write packets. The Transmit Circuit (XMIT) takes packets from the XMB, adds the flag field and converts from the eight bit parallel format to an optical signal. It also processes test packets from the LTB.

A block diagram of the Switch Fabric (SF) is given in Figure 1.20. It contains four major components, a Copy Network, a set of Broadcast and Group Translators, a Distribution Network and a Routing Network. The SF runs at a clock rate of 25 Mbs and has eight bit wide internal data paths. This gives an effective bit rate of 200 Mbs

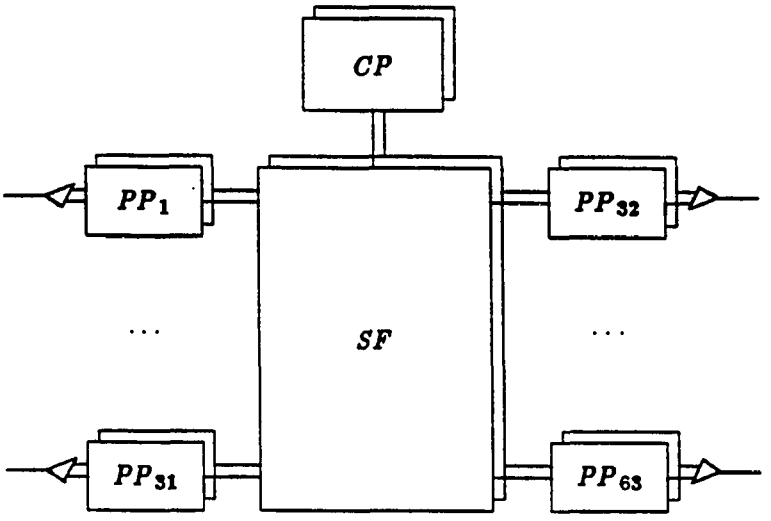


Fig. 1.18. Switch Module

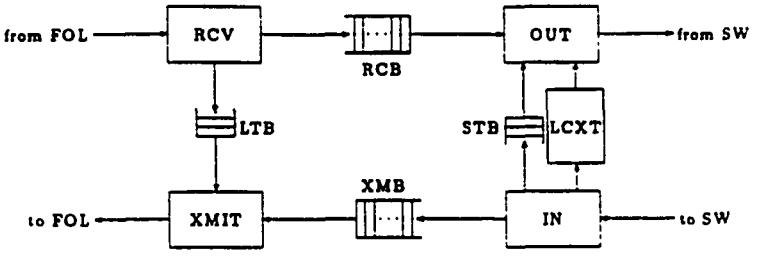


Fig. 1.19. Packet Processor

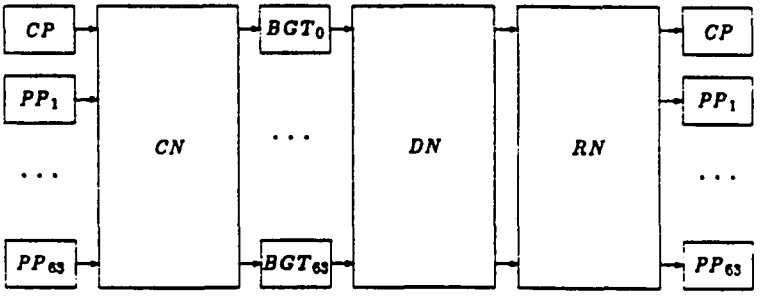


Fig. 1.20. Switch Fabric

on the internal data paths, roughly two times the speed of the FOLs. An occupancy of 80% on the FOLs translates to a 40% occupancy on the internal data paths, which keeps contention and delay low.

When a broadcast packet having k destination passes through the Copy Network (CN), it is replicated so that k copies of that packet emerge from the CN. Point-to-point packets pass through the CN without change. The Broadcast and Group Translators (BGT) perform two translation functions. First, for each arriving broadcast packet they determine the proper outgoing GN (group number) and LCN. Then, they translate the GN to an LN (link number). The Distribution and Routing Networks (DN, RN) move the packets to the proper outgoing PP. The RN is a straightforward binary routing network and the DN is provided to prevent internal congestion within the RN.

A recent experimentation high-speed packet switching system which was designed to transport voice, data, and video all in packetized form and uses a bus architecture is the Packetized Automated Routing Integrated System (PARIS) [27]. Its design philosophy is to use a very simple protocol to achieve low packet delay with an architecture simple enough that it can be implemented even with off-the-shelf components.

The basic structure of the PARIS switch is shown in Figure 1.21. It uses a high-speed bus as shared medium to interconnect input ports to output ports [28]. The maximum bandwidth of the shared bus is taken to be greater than the aggregate capacity of all

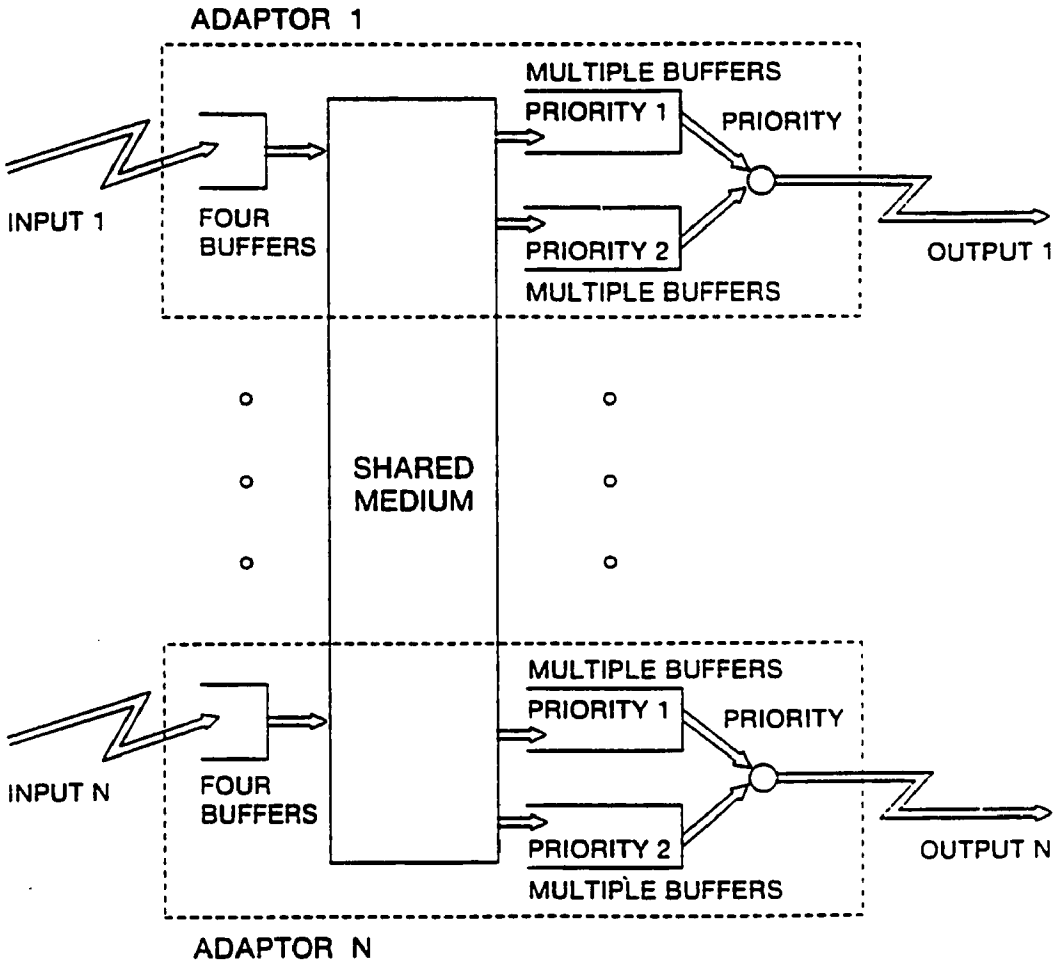


Fig. 1.21. Basic structure of the PARIS switch.

input lines. The switch can handle variably sized packets ranging from 32 bits to a maximum of 8 Kbits. Each input port has a buffer that can hold no more than four packets of maximum size. It is shown that this buffering is adequate to ensure no packet loss, provided a round-robin exhaustive service policy is used to arbitrate the access to the bus. Therefore, the switch is nonblocking. The arbitration protocol is implemented using a fast token-passing algorithm [28]. Each output port has two buffers, one for each priority class. The output buffers are sized such that the packet loss due to a momentary overload situation of an output port is less than 10^{-8}.

1.5.6 SWITCH FABRIC WITH INTERMEDIATE AND OUTPUT QUEUES

Recently Kim and Garcia [29] proposed a self routing switching network consists of 2 x 2 switching elements , distributors, and buffers located between stages and in the output ports. The proposed switching network requires a speed-up factor of two and has $\log_2 N$ stages that moves packets in a store-and-forward fashion. Figure 1.22. Shows block diagram of an 8 x 8 switching network. If the distributors are removed, the linking pattern is the same as that of an 8 x 8 self-routing banyan network. The ith stage switching element (SE) routes the packet to the top if the ith element of the address is 0 or to the bottom if the ith element is 1. The operation of the switching element is synchronous. The clock cycle, T seconds is divided into two subintervals of T_s seconds. Each input port accepts a packet at the beginning of each T second clock cycle. Every T_s second each switching element routes a packet. Thus the SE can route any two input packets to their destinations within one clock cycle.

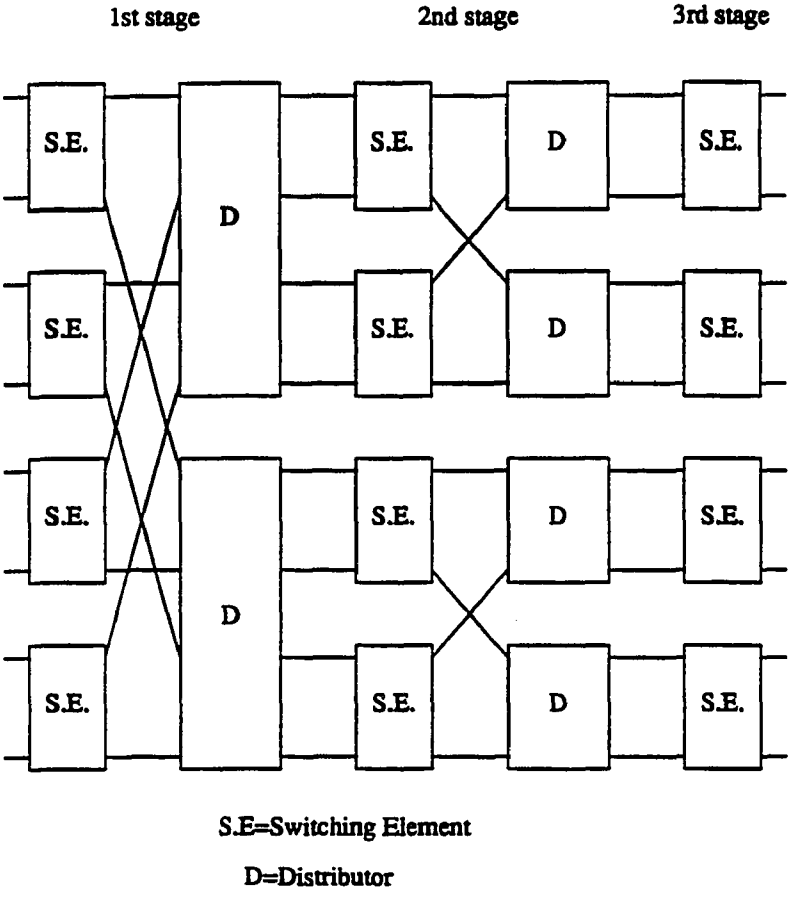


Fig. 1.22. Block diagram of 8 × 8 switching network.

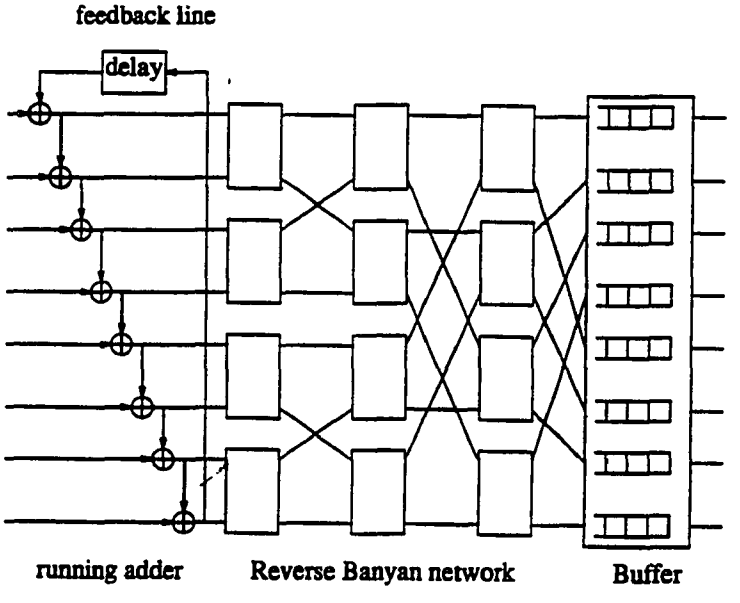


Fig. 1.23. Structure of 8 × 8 distributor.

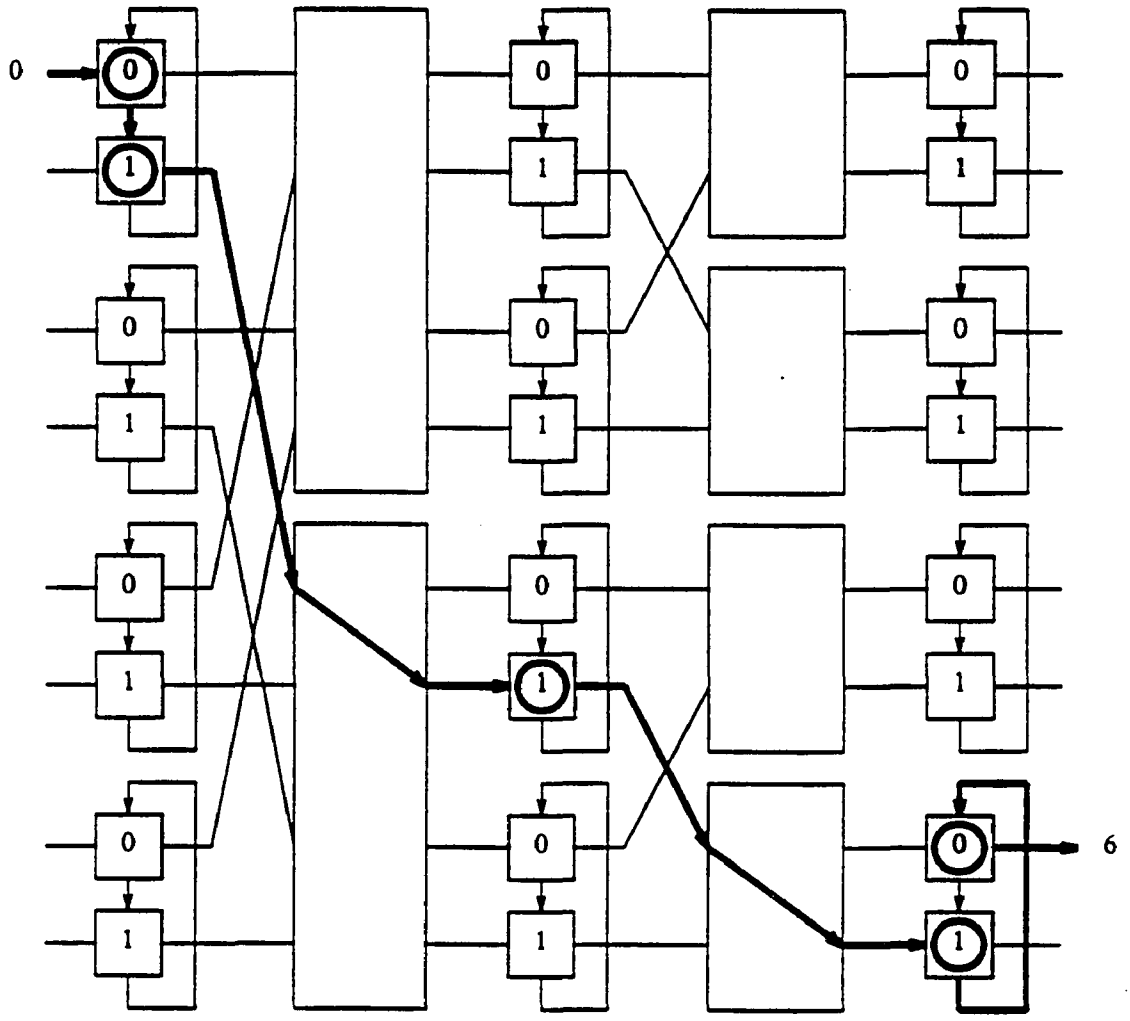


Fig. 1.24. Routing in the switching network.

The distributor evenly distributes the packets to its buffers then uses the buffer space efficiently (see Figure 1.23). Figure 1.24 Shows the routing in the switching network. The bold lines shows the route of a packet from input port 0 to output port 6.

1.6 THESIS OUTLINE

In the previous sections we have seen an overview of the existing switching technologies and modern switches. In the remaining chapters we are proposing and analyzing two broadband packet switches. The first switch proposed in Part I packetizes all traffic into fixed length packets and schedules the queued packets on a demand assignment basis. The second switch, proposed in Part II, can switch circuit traffic and packet traffic in an integrated fashion.

A packet switch based on the idea of assignment on demand has been proposed in chapter 2 and analyzed in chapter 3. Chapter 4 provides the results and conclusion. An overview of some of the existing scheduling algorithms are given in appendix I. The switch model assumes that switch connection can be reconfigured by a control unit according to a switching schedule. The switching schedule may either be provided by an algorithm which will optimize performance or can be a sequence of fixed switching configurations. In both cases, queueing analysis has been carried out. In the analysis, we have assumed that buffer capacity is limited and that the transmission period (frame length) is variable. The analysis is based on a discrete time Markov chain from which state probabilities of the transmission period have been derived. Also, buffer overflow probabilities have been

obtained. Delay versus traffic characteristics with different buffer sizes have been presented and compared for each of the above cases in chapter 4. Performance results indicates that packet delays are 50% to 60% greater when a fixed scheduling algorithm is used compared to when an optimum scheduling algorithm is used. The proposed switch can be used in a variety of applications in terrestrial or satellite networks. The main advantage of this switch model in comparison with other architecture is that in our model the switch connections are software driven and thus can optimize one parameter or another of switch performance.

In chapter 5 an N x N nonblocking time multiplex switch handling circuit and packet traffic is proposed and in chapter 6 the performance of the switch has been examined. We consider a system in which the incoming circuit traffic that can not be routed within a frame size is blocked and the incoming packet traffic which can not be served immediately is queued at the end of the existing input queue. Circuit switched traffic performance of the proposed switch is analyzed in terms of call blocking probability. Packet switched traffic performance is analyzed in terms of mean packet delay. A two dimensional Markov chain is used to model the system. It has been shown, through a special case, that the mean packet queueing delay can be reduced considerably by using movable boundary strategy as compared to fixed boundary strategy. A comparison between the performance of the integrated switching strategy and that of circuit switching strategy has also been provided. It is shown that the circuit blocking probability has improved when integrated with packet traffic.

PART I

2 PACKET SWITCH WITH DEMAND ASSIGNMENT CAPABILITIES

A packet switch based on the idea of assignment on demand has been proposed. The switch model assumes that switch connection can be reconfigured by a control unit according to a switching schedule. The switching schedule may either be provided by an algorithm which will optimize performance or can be a sequence of fixed switching configurations.

2.1 INTRODUCTION

A packet switch is the key element in a high performance data network and has received great attention. (See references [5], [8] and [11]). A number of different approaches have been employed in the design of such a switch. Mainly, a packet switch can either be a single stage or a multistage network, it may have a blocking or nonblocking switch fabric, or it may have a crossbar like or bus type architecture. For example, a banyan type packet switch, proposed in reference [5], is a blocking, multistage interconnect comprised of 2x2 switch elements. Another type, called the "knockout switch" has been presented in reference [8], has a broadcast bus type architecture.

In packet switching however, packet arrivals to the switch are unscheduled, each containing a header bearing address information used to route the packet through the switch. In order to avoid conflict between packets with the same destination, queueing may be

required. Depending on the particular architecture and the speed of the switch fabric, queueing may take place at the input, at the output, or at the internal nodes. For example, a bus matrix switch architecture given in reference [6], has its buffering at internal nodes called cross point memory. Output queueing is possible when the switch output ports run faster than input ports. Such a model has been presented and analyzed in reference [8].

We consider a packet switch which utilizes the method of demand assignment switching. In such a switch, there is a nonblocking crossbarlike switch fabric which can be reconfigured on demand by the switch control unit. See Figure 2.1. It also includes input queueing buffers and a central scheduling unit which assigns the input-output connectivity according to the packet demand in the input buffers. The scheduling unit runs under a scheduling algorithm which resolves conflicts and optimize performance by maximizing throughput. The algorithm may also be a suboptimum one (having reduced computational complexity) or it may simply provide a rotation of predetermined (fixed) sequence of switching configurations. Hence, the presented model can be adopted to different traffic requirements by using the appropriate scheduling algorithm. This model also can be used in a variety of applications in terrestrial or satellite networks.

The problem of finding a demand assignment scheduling algorithm has been studied extensively (see references [30]-[32]), for use in Satellite Switched/ Time Division Multiple Access (SS/TDMA) systems, where optimum and suboptimum algorithms have been presented. The demand assignment approach has also been examined for use in

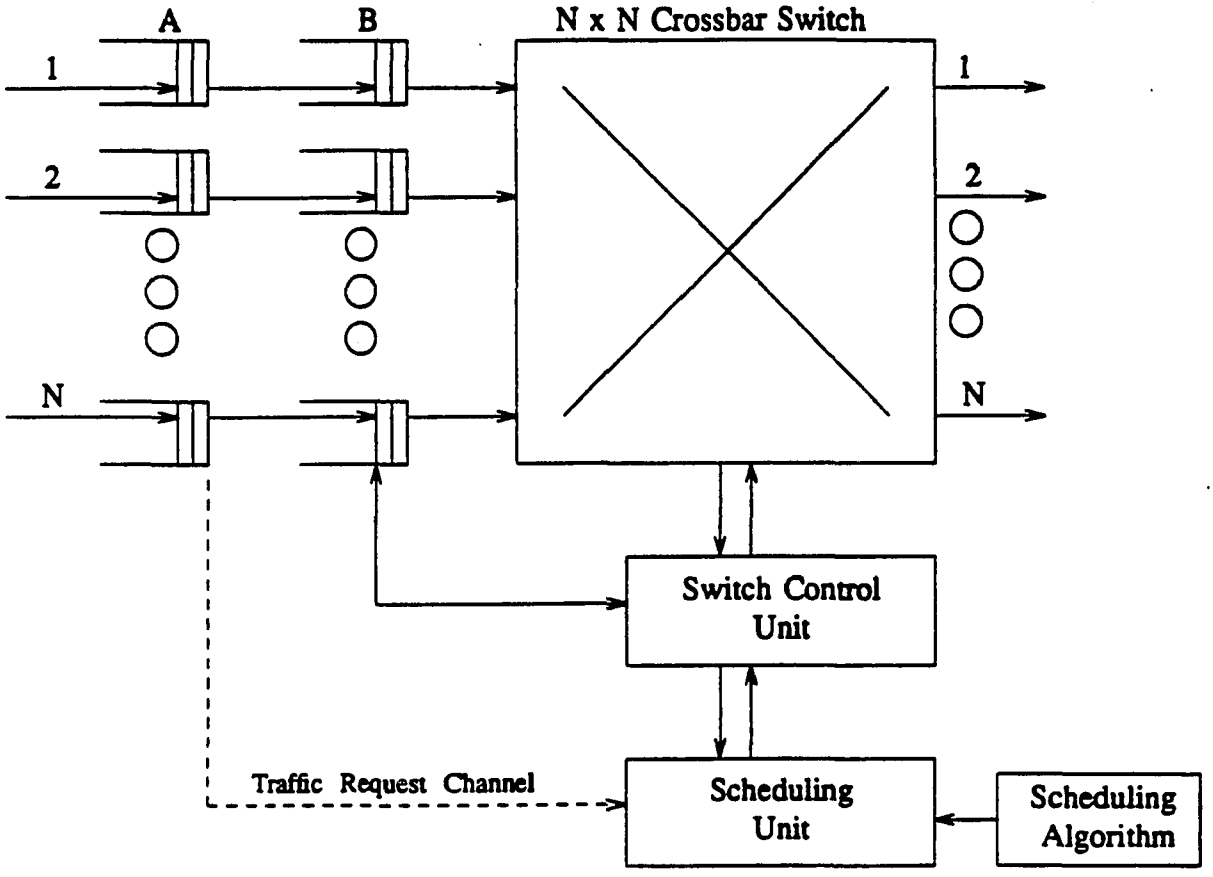
voice traffic blocking system (circuit switching) in references [34] and [35]. In circuit-switching however, the small increase in the traffic load does not justify the use of a computationally expensive optimal algorithm, as shown in references [34] and [35].

The next section provides the model description and chapter 3 provides the delay analysis when an optimum switching algorithm or fixed scheduling algorithm is used. Chapter 4 provides the performance results for both the cases, comparison between them and conclusion. Chapter 5 has the appendices. Appendix I provides a review of some existing scheduling algorithms for demand assignment and in Appendix II a list of symbols used in the analysis is given.

2.2 THE SWITCHING MODEL

The considered packet switch model is shown in Figure 2.1. Conceptually, it consists of the input buffers, the crossbar switch fabric and the units for switch control and scheduling. To describe the operation of this model we assume fixed-length packets arriving to the N input buffers (A). Each arrived packet sends a request via the request channel to the scheduling unit where a list is maintained in matrix form called *Traffic Matrix*, $T = [t_{ij}]$. Each entry t_{ij} in T, represents the traffic demand in packets from input port i to output port j, $1 \leq i \leq N$ and $1 \leq j \leq N$. At the beginning of each *Transmission Period* (TP), the scheduling algorithm is applied in order to decompose the traffic matrix T into a number of switching matrices T_s.

$$T = \sum_{s=1}^{S} T_s \tag{2.1}$$



Conceptual Diagram of the Proposed Packet Switch

Figure 2.1.

T_s has the property of having at most one entry in each row or column and thus corresponds to a switching configuration that routes the traffic via the switch without conflict. The scheduling algorithm is used to program the scheduling unit. It can be an optimal one i.e., minimizing the transmission time, a suboptimal one, or simply a predetermined (fixed) sequence of switching configurations. (See Appendix I)

After the scheduling algorithm has been executed, packets in buffers (A) will be transferred to buffers (B). The switch control unit will then apply each scheduling matrix T_s to reconfigure the NxN switch fabric and route the traffic from input to output ports.

We define the length of a *Transmission Period* (TP) or *Transmission Length* (TL) as the time needed to switch the total amount of traffic in the traffic matrix. During a TP and while packets departing from buffers (B) to be switched, new packets arrive to the input buffers (A) to form the next traffic matrix which will be transmitted on the next TP.

The Transmission Length depends on the switching algorithm. The minimum possible TL is equal to the *Maximum Line Sum* Z, of the traffic matrix T. Z is defined by:

$$Z = \max\{r_1, r_2, \ldots, r_N, c_1, c_2, \ldots, c_N\} \tag{2.2}$$

where r_1, r_2, ..., r_N are the sum of the elements in a row (row sum) and c_1, c_2, ..., c_N are the column sums in T. The optimal algorithm always achieves TL equal Z, while other algorithms may produce transmission lengths L = aZ, with $a > 1$. An example is given in Figure 2.2. Figure 2.2.a shows a traffic matrix T from input ports (A, B, C and D) to

output ports (A', B', C', and D'). When the scheduling algorithm is applied to the traffic matrix, it is decomposed into a number of switching matrices. The switching matrix has atmost one nonzero entry in each row or column. The switching matrix corresponds to one switching mode in a transmission period and its maximum element determines the mode duration. The transmission period is the sum of the mode durations. Figure 2.2.b shows a possible transmission schedule for the traffic matrix shown in Figure 2.2.a. Here the transmission length is 88 > 77 which is the maximum line sum (critical line). The minimum transmission length possible is the maximum line sum. Figure 2.2.c shows a set of switching matrices which achieves the minimum transmission length.

To obtain the optimum scheduling switching matrices as shown in Figure 2.2.c choose the largest entry from the critical line of the T matrix in Figure 2.2.a. Obtain 3 more entries for the switching matrix such that there is only one entry in each row or column. Subtract this switching matrix from T to obtain the new traffic matrix T. Check for the critical line and add back enough numbers to keep the same critical line (take from the switching matrix). Also the entry in the switching matrix belonging to the critical line should be maximum, if not move the extra numbers back to T. Repeat all these steps untill T has no more nonzero elements. Please refer to [30]-[32] for more details.

T =

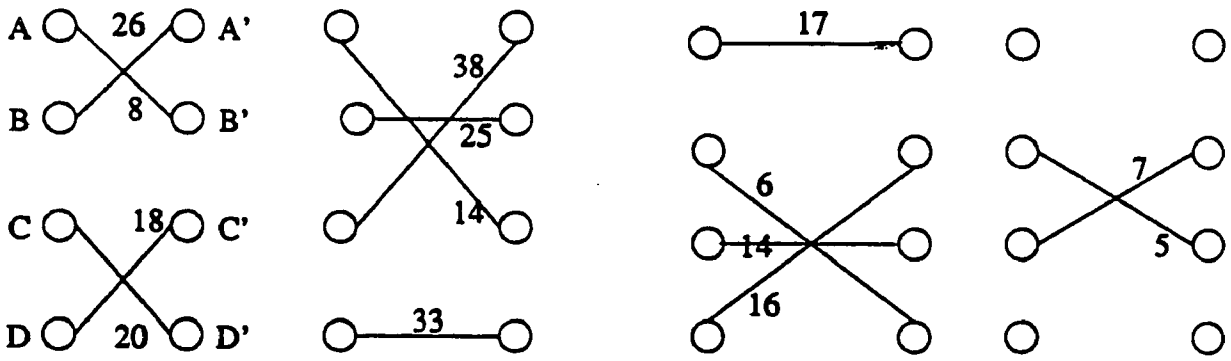
Input \ Output	A'	B'	C'	D'	row sum
A	17	8	38	0	63
B	26	25	7	6	64
C	14	5	14	20	53
D	0	16	18	33	67
column sum	57	54	77	59	247 ← total traffic

↑ critical line

Figure 2.2a. Traffic Matrix T

$$T = \begin{bmatrix} & 8 & & \\ 26 & & & \\ & & & 20 \\ & & 18 & \end{bmatrix} + \begin{bmatrix} & & 38 & \\ & 25 & & \\ 14 & & & \\ & & & 33 \end{bmatrix} + \begin{bmatrix} 17 & & & \\ & & & 6 \\ & & 14 & \\ & 16 & & \end{bmatrix} + \begin{bmatrix} & & & 0 \\ & & 7 & \\ & 5 & & \\ 0 & & & \end{bmatrix}$$

$$|T_1| = 26 \qquad |T_2| = 38 \qquad |T_3| = 17 \qquad |T_4| = 7$$



Transmission Length L = 26 + 38 + 17 + 7 = 88 > 77 S=N=4

Figure 2.2b. A Possible Transmission Schedule for Matrix T

$$T = \begin{bmatrix} & & 26 & \\ 26 & & & \\ & & & 20 \\ & 16 & & \end{bmatrix} + \begin{bmatrix} & & 12 & \\ & 12 & & \\ 12 & & & \\ & & & 12 \end{bmatrix} + \begin{bmatrix} 14 & & & \\ & 13 & & \\ & & 14 & \\ & & & 14 \end{bmatrix} +$$

$$+ \begin{bmatrix} & 8 & & \\ & & & 6 \\ 2 & & & \\ & & 18 & \end{bmatrix} + \begin{bmatrix} 3 & & & \\ & & 7 & \\ & 5 & & \\ & & & 7 \end{bmatrix}$$

$$L = 26 + 12 + 14 + 18 + 7 = 77 = Z \qquad S = 5$$

Figure 2.2c. An optimum transmission schedule of matrix T, i.e., minimizing transmission length.

3 DELAY ANALYSIS

In this chapter the analysis of the switch proposed in chapter 2 is given. In section 3.1 we have the analysis when an optimum scheduling algorithm is used and in section 3.2 we have the analysis when a fixed scheduling algorithm is used.

3.1 ANALYSIS FOR OPTIMUM SCHEDULING

We assume that the input channels are time slotted. Each packet has fixed length equal to one time slot. Packets arriving to the N input buffers are governed by an independent and identical Bernouli process. Specifically, in any given time slot, the probability that a packet will arrive to a particular input is σ. Each packet has equal probability of being addressed to any given output and successive packets are independent. The input buffers are assumed to have limited capacity. The size of each input buffer (A) and (B) are M/N, where M mod N is zero.

This analysis aims in obtaining the queueing delays by finding the steady state distribution of the transmission length L, which is obtained from the solution of a discrete time Markov Chain. Considering our switching system in steady-state operation, we observe that the total number of packets, that arrived during the current transmission period TP(k-1) will form the traffic matrix that will be transmitted on the next transmission period TP(k) (see Figure 3.1). Hence, the length of TP(k), L_k, depends on the length of TP(k-1), L_{k-1}.

Let $h(L_k/L_{k-1}=L)$ be the conditional probability that a TP(k) has length L_k, given TP(k-1) has length L. In general any scheduling algorithm generates transmission length L_k

$$L_k = aZ \tag{3.1}$$

where Z is a random variable (r.v.) representing the maximum line sum in T, and a is a constant ($a \geq 1$) depending on the algorithm used. An optimum algorithm always achieves $L_k = Z$

$$Z = \max\{R, C\} \tag{3.2-a}$$

where

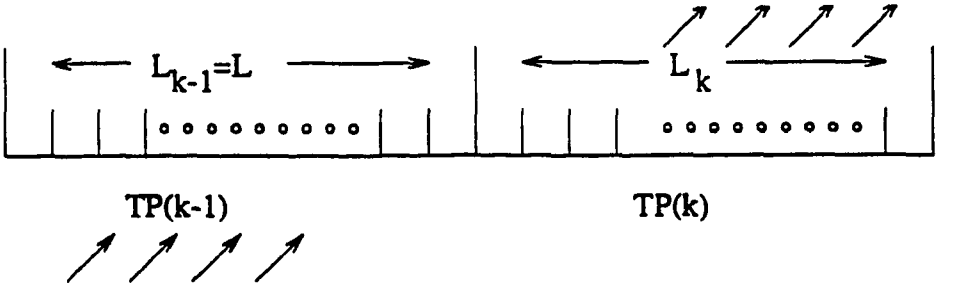
$$R = \max\{r_i\} \qquad 1 \leq i \leq N \tag{3.2-b}$$

$$C = \max\{c_j\} \qquad 1 \leq j \leq N \tag{3.2-c}$$

In order to obtain the distribution of $h(L_k/L)$, first we need to obtain the distribution of r.v.s R and C and the distribution of r.v. Z. In obtaining the distribution of r.v.s Z, R and C we make use of equations (3.2-a), (3.2-b) and (3.2-c) respectively.

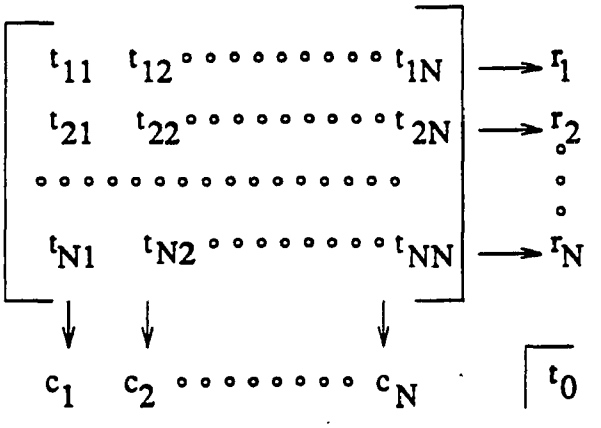
The row sum r_i equals to the number of arrivals to an input buffer (A) during TP(k-1) of length L at input i (see Figure 3.2). Each input buffer (A) is assumed to have a maximum capacity of M/N packets. Hence

$$Pr[r_i = l] = \frac{1}{b(r_i)}\begin{bmatrix} L \\ l \end{bmatrix}\sigma^l(1-\sigma)^{L-l} \tag{3.3}$$



Packets that arrive at transmission period TP(k-1) will depart at TP(k).

Figure 3.1.



$Z = \max\{R,C\}$ $\quad R = \max\{r_i\}$ $\quad C = \max\{c_j\}$

$0 \le R \le L$ $\quad 0 \le C \le NL$ $\quad 0 \le Z \le NL$

Figure 3.2.

for $l = 0,1,2,...L$, where $b(r_i)$ is to normalize r.v. r_i, when $M/N \le L \le M$ because r_i is limited to the maximum value M/N by the buffer capacity per input port. If the input buffer is not limited to M/N then r_i can go upto L, if a packet arrives in each slot. Therefore the normalization constant $b(r_i)$ is given by:

$$b(r_i) = \sum_{l=0}^{l_m} \binom{L}{l} \sigma^l (1-\sigma)^{L-l} \quad where \quad l_m = \min\{L, M/N\} \qquad (3.3-a)$$

Now, since all r_i $1 \le i \le N$ are independent, identically distributed r.v.s, the distribution of the largest of them R, is given by:

Pr[R = max{r_i}] = Pr[one r_i = r]Pr[all other r_i's < r] + Pr[two r_i = r]Pr[all other r_i's , r] + ... + Pr[(n-1) r_i = r]Pr[all other r_i < r] + Pr[n r_i = r]

$$Pr[R = \max\{r_i\}] = \sum_{i=1}^{N} \left\{ \binom{N}{i} [Pr(r_i = r)]^i [Pr(r_i < r)]^{N-i} \right\} \qquad (3.4-a)$$

$$or \quad Pr[R = r] = \sum_{i=1}^{N} \left\{ \binom{N}{i} \left[\frac{1}{b(r_i)} \binom{L}{r} \sigma^r (1-\sigma)^{L-r} \right]^i \left[\sum_{l=0}^{r-1} \frac{1}{b(r_i)} \binom{L}{l} \sigma^l (1-\sigma)^{L-l} \right]^{N-i} \right\} \qquad (3.4)$$

The probability of l arrivals to input i for a particular destination j, during TP(k-1) of length L is:

$$Pr[t_{ij} = l] = \frac{1}{b(t_{ij})} \binom{L}{l} (\sigma/N)^l (1-\sigma/N)^{L-l} \qquad (3.5)$$

where $b(t_{ij})$ normalizes $Pr(t_{ij}=l)$ to the maximum value of M/N.

$$b(t_{ij}) = \sum_{l=0}^{l_m} \begin{bmatrix} L \\ l \end{bmatrix} (\sigma/N)^l (1-\sigma/N)^{L-l} \qquad (3.5-a)$$

The total traffic t_0 is is the sum of traffic from every input to every output. i.e. t_0 is the sum of all entries in traffic matrix T. Hence

$$t_0 = \sum_{i=1}^{N} r_i = \sum_{j=1}^{N} c_j = \sum_{j=1}^{N} \sum_{i=1}^{N} t_{ij} \qquad (3.6)$$

Since t_0 is the sum of independent random variables r_i's, t_0 is also binomial distributed with mean $NL\sigma$ and variance $NL\sigma(1-\sigma)$. Hence:

$$Pr[t_0 = l] = \frac{1}{b(t_0)} \begin{bmatrix} NL \\ l \end{bmatrix} \sigma^l (1-\sigma)^{NL-l} \qquad (3.7)$$

for $0 \le l \le Nl_m$ *where* $l_m = \min\{L, M/N\}$ and $b(t_0)$ normalizes $Pr(t_0=l)$ to the maximum value M.

$$b(t_0) = \sum_{l=0}^{Nl_m} \begin{bmatrix} NL \\ l \end{bmatrix} \sigma^l (1-\sigma)^{NL-l} \qquad (3.7-a)$$

In deriving the distribution of column sums c_j the assumption is made that each packet has equal probability 1/N of being addressed to any given output, and successive packets

are independent. Now, since t_0 is the total number of packets arrived during TP(k-1) = L, the probability of having l packets destined for the output j is given by:

$$Pr[c_j = l/t_0] = \begin{bmatrix} t_0 \\ l \end{bmatrix} (1/N)^l (1-1/N)^{t_0-l} \tag{3.8}$$

Since $l \le t_0 \le Nl_m$, the unconditional probability $Pr[c_j]$ is given by:

$$Pr[c_j = l] = \sum_{t_0=l}^{Nl_m} \left\{ \left[\begin{bmatrix} t_0 \\ l \end{bmatrix} (1/N)^l (1-1/N)^{t_0-l} \right] Pr[t_0] \right\} \tag{3.9}$$

The distribution of the largest of them $C = \max\{c_j\}$ is given by:

$$Pr[C = c] = \sum_{j=1}^{N} \left\{ \begin{bmatrix} N \\ j \end{bmatrix} [Pr(c_j = c)]^j \left[\sum_{l=0}^{c-1} Pr(c_j = l) \right]^{N-j} \right\} \tag{3.10}$$

Finally, the distribution $Z = \max\{R,C\}$ is given by:

$$Pr[Z = z] = \begin{Bmatrix} Pr(R = z)Pr(C \le z) + Pr(C = z)Pr(R \le z) & for \quad 0 \le Z \le l_m \\ Pr(C = z) \quad for \quad l_m < Z \le Nl_m \end{Bmatrix} \quad (3.11-a)$$

$$Pr[Z = z] = \begin{Bmatrix} Pr(r = z) \sum_{c=0}^{z} Pr(C = c) + Pr(C = z) \sum_{r=0}^{z} Pr(R = r) & for \quad 0 \le Z \le l_m \\ Pr(C = z) \quad for \quad l_m < Z \le Nl_m \end{Bmatrix} \quad (3.11)$$

because $0 \le R \le l_m, \quad 0 \le C \le Nl_m$ and $l_m = \min\{L, M/N\}$

The distribution of the transition probabilities for the case of having optimum scheduling algorithm will be as in equation (3.11). Hence:

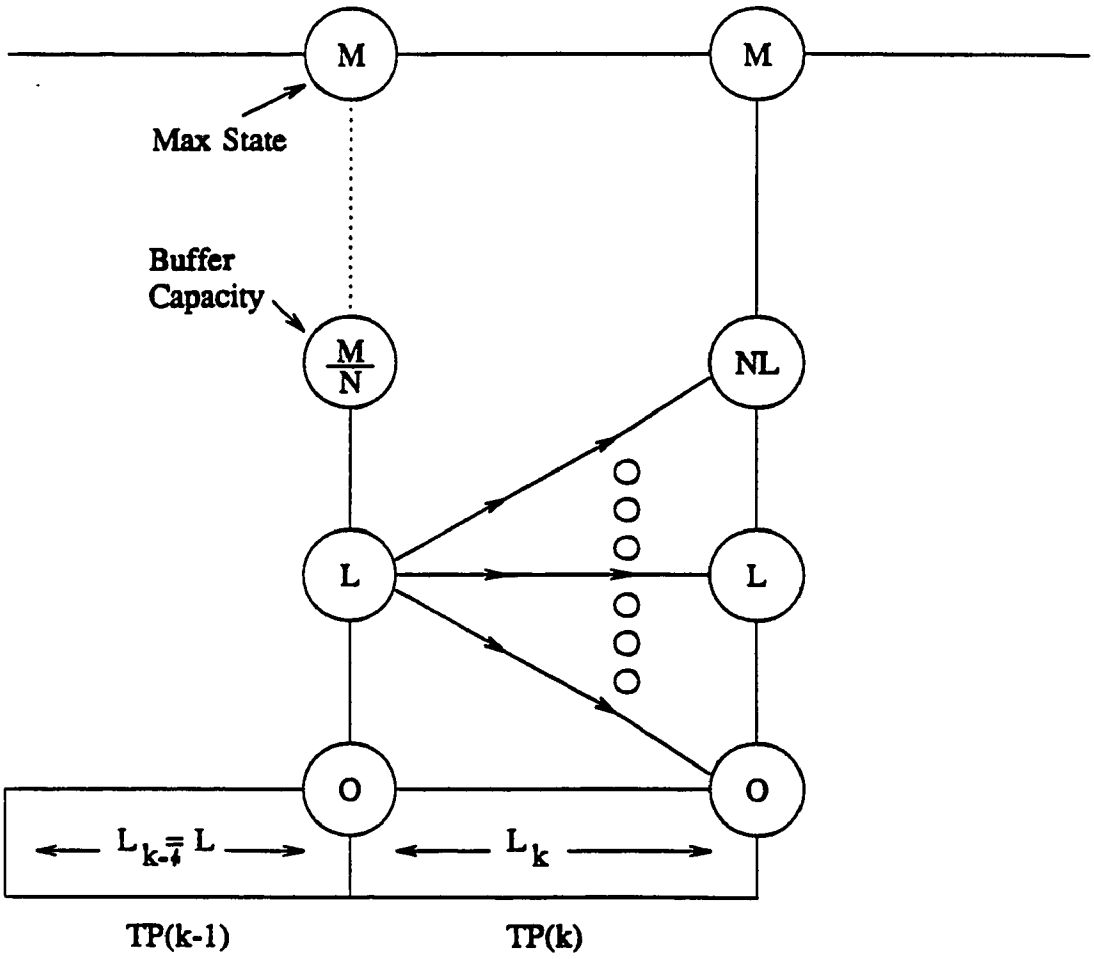
$$h(L_k = Z / L_{k-1} = L) = Pr[Z = z] \tag{3.12}$$

System Markov Chain

Considering the transmission length L as the state of the system, we obtain the state probabilities $P_L = Pr[L=l]$ in steady state operation, from the solution of the following discrete time Markov Chain, (see Figure 3.3).

$$P_L = P_L H \tag{3.13}$$

where the vector $\mathbf{P_L}$ represents the distribution of L, $\mathbf{P_L} = (P_0, P_1, ..., P_L, ..., P_M)$ and $\mathbf{H} = [h_{nm}]$ is the matrix of transition probabilities. Each entry h_{nm} in **H** represents the transition probability from state $L_{k-1} = n$ to state $L_k = m$; i.e., $h_{nm} = h(L_k=n/L_{k-1}=m)$ is given by equation (3.12). The maximum number of states is M, which results from having a maximum buffer capacity M/N packets per input port.



if $0 \leq L \leq M/N$ then $0 \leq L_k \leq NL$
if $M/N \leq L \leq M$ then $0 \leq L_k \leq M$

Markov Chain

Figure 3.3.

Average Delay

Let $\overline{L_0}$ be the average transmission length generated by the optimum algorithm; then $\overline{L_0} = \overline{Z}$ where $\overline{Z}$ is the average maximum line sum.

$$\overline{L_0} = \sum_{L=0}^{M} LP_L \tag{3.14}$$

The total average delay D may be expressed as the sum:

$$D = D_w + D_s \tag{3.15}$$

where D_w is the time elapsed from the moment of a packet arrival until the end of TP(k-1) (waiting delay), and D_s is the time elapsed from the beginning of TP(k) until departure (service delay). The assumption is made that on the average, a packet arrives in the middle of a TP i.e., $D_w = 1/2\overline{L_0}$. The average service delay D_s, has been defined by equation (I-2) in Appendix I. The value of D_s actually depends on the algorithm used. An algorithm which minimizes the service delay D_s also minimizes the transmission length L. In this case, $D_s \leq 1/2\overline{L_0}$ (see reference [30]). On the average however, $D_s = 1/2\overline{L_0}$. Hence, $D = \overline{L_0}$.

In the case where a suboptimum algorithm is used $\overline{L} = a\overline{L_0}$ with $1 < a \leq N$.

Buffer Overflow Probability (P_B) and Throughput (γ)

Since the input buffers (A) are limited to M/N packets, overflow may occur when TP length is $L > M/N$; then any more than M/N arrivals will be turned away. Hence, the buffer-overflow probability P_B will be:

$$P_B = Pr[M/N < r_i = l \leq L]Pr[L > M/N] \qquad (3.16-a)$$

$$\text{or} \quad P_B = \left[\sum_{l=M/N+1}^{L}\begin{bmatrix} L \\ l \end{bmatrix}\sigma^l(1-\sigma)^{L-l}\right]\left[\sum_{L=M/N+1}^{M} P_L\right] \qquad (3.16)$$

Finally, the throughput γ is given by:

$$\gamma = \sigma(1-P_B) \qquad (3.17)$$

where σ is the probability of a packet arrival in a time slot (offered load).

3.2 ANALYSIS FOR FIXED SCHEDULING

In fixed scheduling, a predetermined sequence of switching configurations rotates for the duration of each transmission period. An example of such a sequence of N switchings is shown in Figure 2.2.a. Let T represents a traffic matrix in terms of a possible switching schedule as shown in the example below:

$$T = \begin{bmatrix} t_1(1) & t_2(1) & t_4(1) & t_3(1) \\ t_2(2) & t_1(2) & t_3(2) & t_4(2) \\ t_4(3) & t_3(3) & t_2(3) & t_1(3) \\ t_3(4) & t_4(4) & t_1(4) & t_2(4) \end{bmatrix}$$

In the above representation elements $t_x(y)$ for y = 1, ..., N have no common rows or columns and thus may belong to the same switching matrix T_x, where x is the switching matrix number and y is the element number in that switching matrix. Hence, using the above mask on every traffic matrix T, we extract N switching matrices $T_x = [t_x(y)]$ for x = 1, ..., N. The switching duration of each $|T_x|$ is then given by:

$$V_x = |T_x| = \max\{t_x(1), t_x(2), \ldots, t_x(N)\} \tag{3.18}$$

The probability distribution of r.v. V_x can be found using equation (3.4-a)

$$Pr[V_x = \alpha] = \sum_{i=1}^{N} \left\{ \begin{bmatrix} N \\ i \end{bmatrix} [Pr[t_x(y) = \alpha]]^i \left[\sum_{l=0}^{\alpha-1} Pr[t_x(y) = l] \right]^{N-i} \right\} \tag{3.19}$$

where $0 \le V_x \le l_m$. The distribution of $t_x(y)$ is given by equation (3.5). Then, the transmission length V, generated by fixed scheduling is given by:

$$V = V_1 + V_2 + \ldots + V_N \tag{3.20}$$

Now, considering a Markov chain with the transmission length V as the state, the

transition probabilities from state L in TP(k-1) to state V in TP(k) can be obtained as follows:

$$h(L_k = V/L_{k-1} = L) \quad = \quad Pr[V_1 = \alpha]*Pr[V_2 = \alpha]*\ldots*Pr[V_n = \alpha] \qquad (3.21)$$

In the above equation (*) represents a convolution summation. This results from equation (3.20) in which V is a sum of independent and identically distributed random variables V_x, x = 1, ..., N, each having mean value p which is given below:

$$p = E[V_x] \quad = \quad \sum_{\alpha=0}^{l_m} \alpha Pr[V_x = \alpha] \qquad (3.22)$$

Following equation (3.21), the distribution of r.v. V may be approximated as in equation (3.23), with the use of central limit theorem (see [36]).

$$Pr[V = \alpha] \quad = \quad \frac{1}{b(v)} \begin{bmatrix} NL \\ \alpha \end{bmatrix} (p/L)^{\alpha} (1 - p/L)^{NL-\alpha} \qquad (3.23)$$

for $0 \le \alpha \le NL_m$ and $l_m = \min\{L, M/N\}$; where b(v) normalizes the above distribution to the maximum value of $\alpha = Nl_m$. The transition probabilities then are given by equation (3.23), i.e., $h(L_k = v/L_{k-1} = L) = Pr[v = \alpha]$.

To find the distribution of the transmission length L, Pr[L=l], we solve the Markov chain of equation (3.13); $P_L = P_L H$. Hence, the average transmission length $\overline{L_f}$ under fixed

scheduling is:

$$\overline{L_f} = \sum_{l=0}^{M} l Pr[L = l] \tag{3.24}$$

The resulting overhead $a_f = \overline{L_f}/\overline{Z}$ (where Z is the maximum line sum) may be as large as N, i.e.,$1 \leq a_f \leq N$.

4 PERFORMANCE RESULTS AND CONCLUSION

In this chapter we discuss the performance of the switch proposed in chapter 2.

4.1 RESULTS AND DISCUSSIONS

The performance characteristics which result from the analysis of chapter 3 is shown in Figures 4.1 to 4.5. (In the Figure numbers (a) corresponds to the case where the optimum algorithm has been used while (b) corresponds to the fixed scheduling case)

Figures 4.1a and 4.1b demonstrate the delay versus the packet arrival per input port σ (offered load) for different values of the switch size N but with fixed buffer size per input port M/N = 6. For $\sigma = 0.5$ it shows that the delay for the optimum scheduling is 50% to 60% smaller than for the fixed scheduling. It also shows, that the maximum delay (*for* $\sigma \rightarrow 1$) saturates to a value which is in the order of switch size N.

Figures 4.2a and 4.2b demonstrate the throughput versus the offered load σ characteristics, while Figures 4.3a and 4.3b the delay versus the throughput. The above characteristics are plotted for different values of the switch size N, but with fixed value of buffer size per input port M/N = 6. As it is indicated the maximum throughput occurs at about $\sigma = 0.6$ for the optimum case and for fixed scheduling at about $\sigma = 0.5$ for $N > 2$, further increase of σ beyond these values results in performance degradation (see Figures 4.2a and 4.2b). If however, the buffer size is increased these values also increase. The value

of the maximum throughput depends on N and M. That is, decreasing with increasing switch size N (see Figures 4.2 and 4.3), but is increasing with increasing buffer size M/N (as shown in Figure 4.5 for the optimum case). For example the maximum throughput for N = 5 and buffer size M/N = 6 is $\gamma = 0.55$ for optimum scheduling and $\gamma = 0.45$ for fixed scheduling. As shown in Figure 4.6, the value of buffer size per input port M/N should be M/N > 10 in order to reduce the buffer overflow probability at acceptably low values ($P_B < 10^{-4}$)

4.2 CONCLUSION

The proposed Packet Switch provides a flexible solution to a variety of applications; it can be used in a local environment or in satellite networks, e.g., SS/TDMA system. The main advantage of this switch model in comparison with other architectures given in references [5], [8] and [11], is that switch connections are software driven. Hence, the switch hardware can be independent from its software (i.e., the switching algorithm) and thus different switching algorithm may reside in the Scheduling Unit which will optimize one parameter or another of switch performance. Also, the switch can be adapted to different traffic statistics by choosing the appropriate scheduling algorithm that will optimize the performance in each case.

Performance results indicate that the demand assignment switch can provide high throughput and low delays when an optimum switching algorithm is used (i.e., this algorithm maximizes throughput by minimizing the length of the transmission period). Performance comparisons are made with the case of fixed scheduling in which packet

delays are 50% to 60% greater. Fixed switching however, do not require computing power. Similar comparisons can also be made with other algorithms, suboptimum or locally optimum, i.e., optimizing performance on a slot-by-slot basis.

By using N subqueues from each input we are avoiding the HOL blocking. This way the switch can obtain better performance than regular input queueing switch.

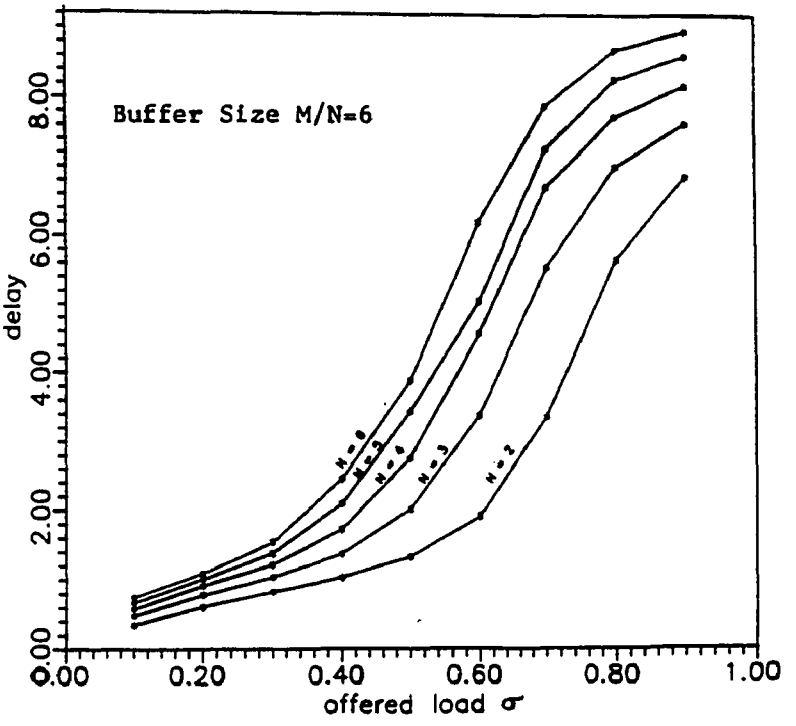


Fig. 4.1a. The delay in time slots versus the offered load σ for the case of optimum switching algorithm.

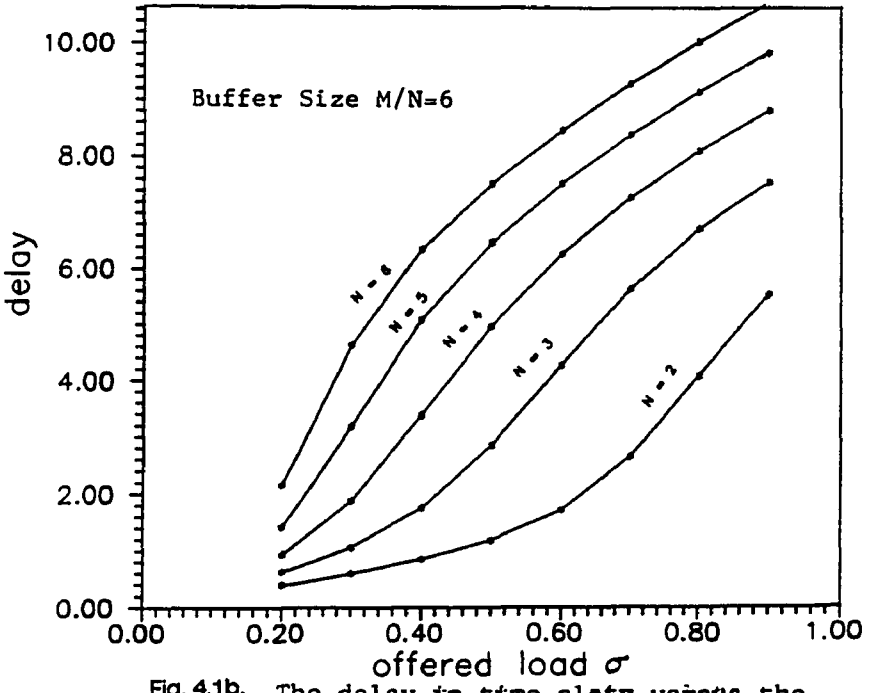


Fig. 4.1b. The delay in time slots versus the offered load σ for the case of fixed scheduling.

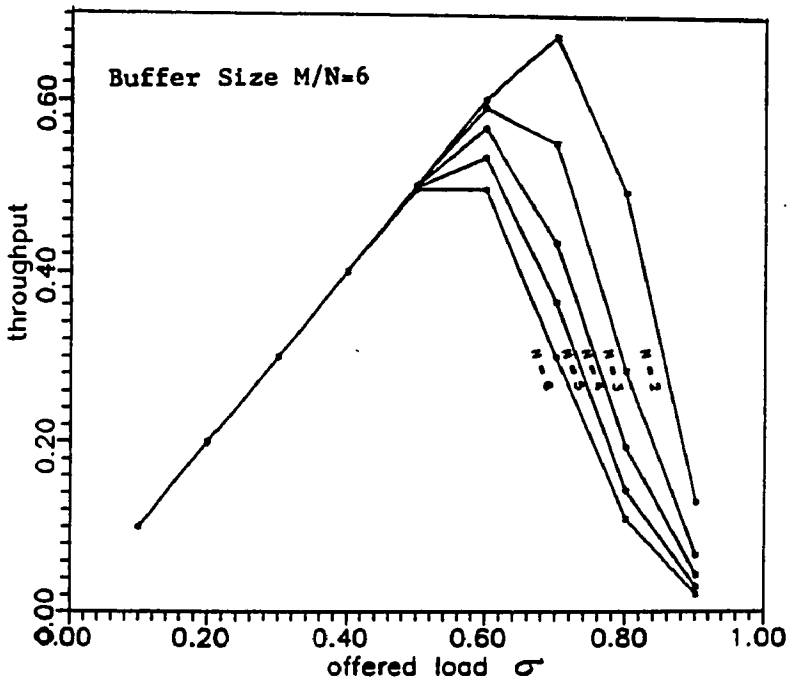


Fig. 4.2a. The Throughput versus the offered load o for the case of optimum switching algorithm.

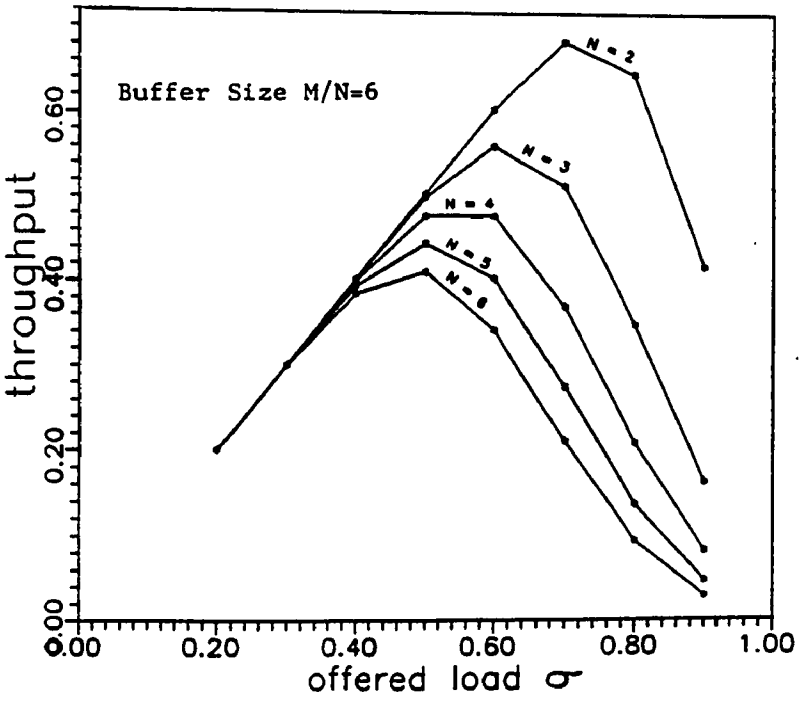


Fig. 4.2b. The Throughtput versus the offered load σ for the case of fixed scheduling.

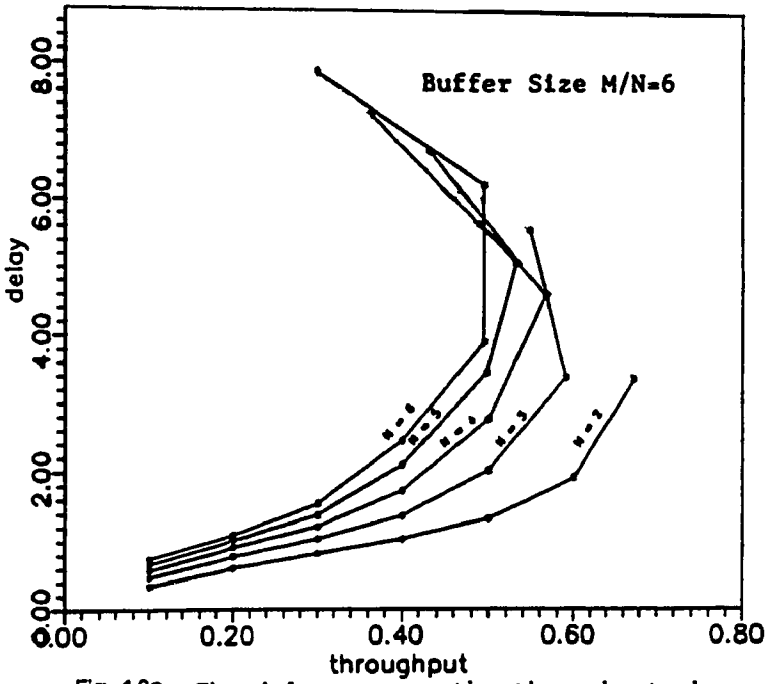


Fig. 4.3a. The delay versus the throughput characteristics for the case of optimum switching algorithm.

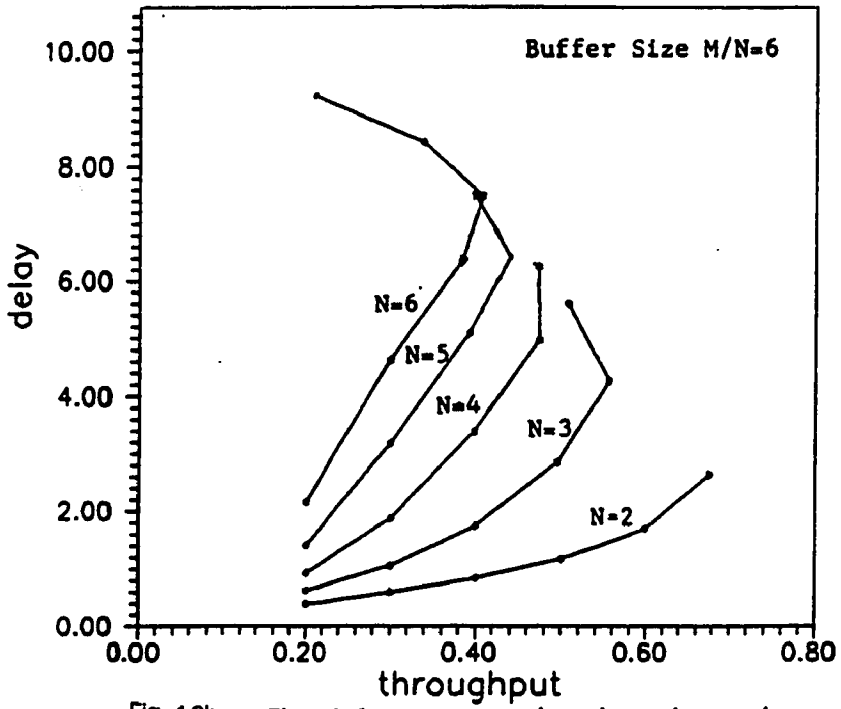


Fig. 4.3b. The delay versus the throughput characteristics for the case of fixed scheduling.

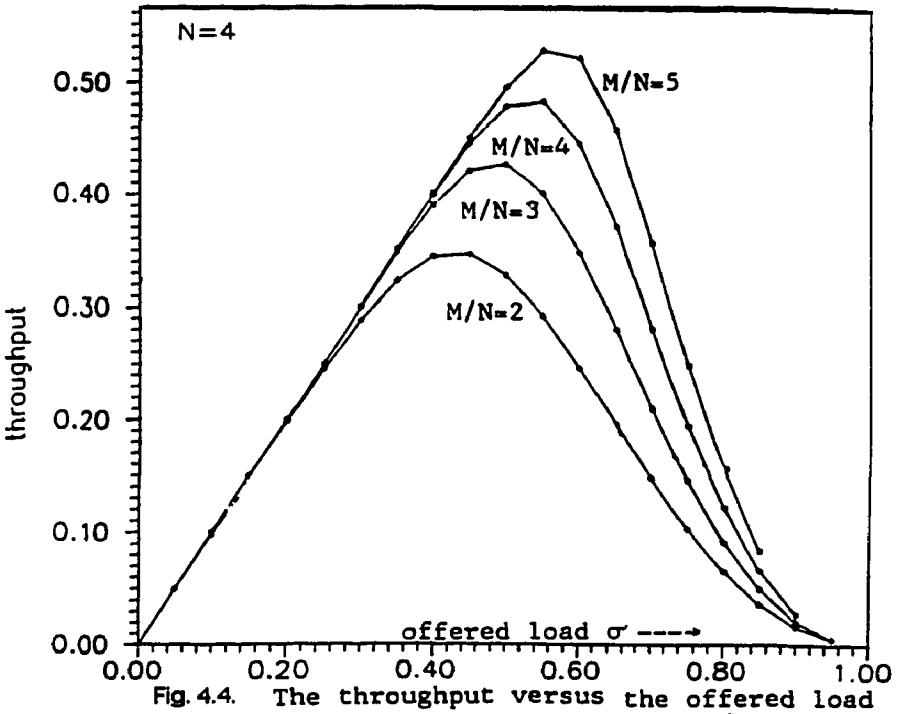


Fig. 4.4. The throughput versus the offered load for different values of input buffer M/N..

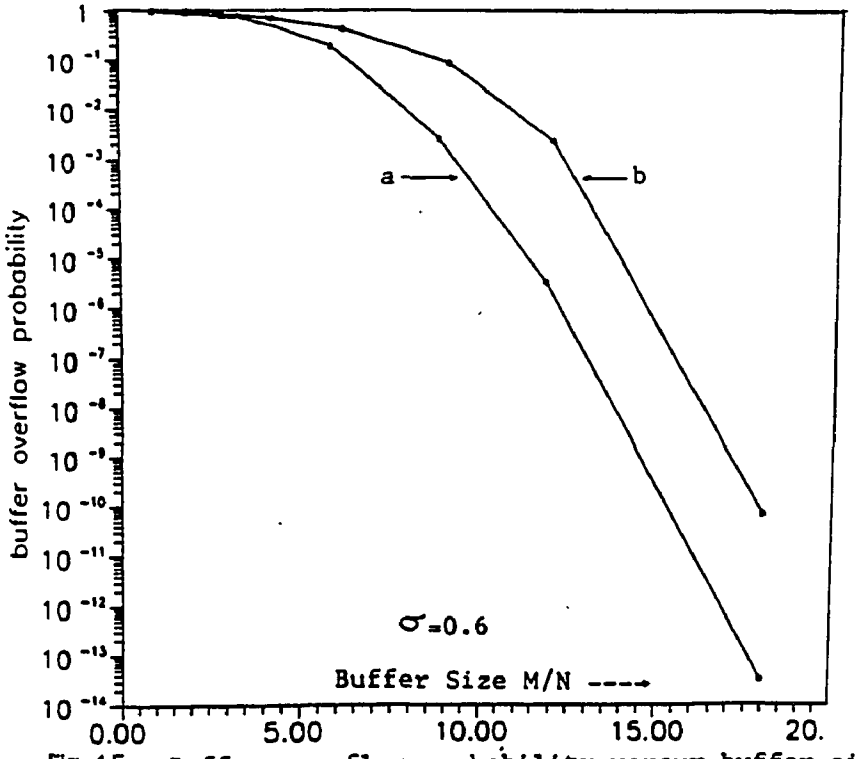


Fig. 4.5. Buffer overflow probability versur buffer size for the cases of optimum scheduling (a) and fixed scheduling(b)

PART II

5 AN INTEGRATED CIRCUIT-PACKET TIME MULTIPLEXED SWITCH WITH RANDOM SCHEDULING

In this chapter we propose an N x N nonblocking time multiplex switch handling circuit and packet traffic. We consider a system in which the incoming circuit traffic that can not be routed within a frame size is blocked and the incoming packet traffic which can not be served immediately is queued at the end of the existing input queue. Circuit switched traffic performance of the proposed switch is analyzed in terms of call blocking probability. Packet switched traffic performance is analyzed in terms of mean packet delay. A two dimensional Markov chain is used to model the system.

5.1 INTRODUCTION

In this paper we are integrating circuit and packet switching using a time-multiplex switch. A time multiplex switch routes time multiplexed traffic from its inputs to its outputs. Switching is accomplished by dividing time into frames of duration $L\tau$ where L is an integer and τ is the duration of the elementary time unit, i.e., *'the time slot'* [35]. In circuit switching a channel is maintained end to end for the duration of a call using the time division multiplex technique. A user requiring one fixed rate circuit is assigned one free time slot per frame in which to transmit a fixed length information packet. The switch then routes the packet to the appropriate output during that time slot. Circuit switching is attractive for high and deterministic data volumes, e.g. voice and bulk data.

In packet switching however, packet arrivals to the switch are unscheduled. In order to avoid conflict between packets with the same destination, queueing is required. Depending on the particular requirements and the speed of the switch fabric, queueing may take place at the input, output or at internal nodes. A number of different queueing strategies have been examined in [15] each resulting in different throughputs. For bursty data traffic packet switching is preferred since it offers improved channel utilization.

An integrated network with combined circuit and packet switching capability would therefore enable a diversity of user requirements to be met simultaneously. With both circuit and packet switching capabilities, hybrid switching will effectively accommodate any mix of traffic. Also it is easier to update the existing telephone network to handle hybrid switching compared to fast packet switching or burst switching. This paper presents an analysis of a multichannel TDMA protocol. We consider a system in which incoming circuit traffic that can not be routed within a frame size is blocked. The incoming packet traffic which can not be served immediately is queued at the end of the existing input queue. If there are N inputs and N outputs then there will be N input queues. Each input queue has N sub queues going to N different outputs. The service is FCFS for each sub queue and not necessarily FCFS for the input queue as a whole. Infinite buffer size is assumed in the analysis in the next chapter.

The problem of switching integrated circuits and packets via the same NxN nonblocking switch can be defined in terms of two main criteria:

(a) Circuit-packet integration strategy: Among the several proposals for hybrid-switched

multiplexing in integrated voice and data systems, the one that has received considerable attention is the Slotted Envelope Network (SENET) concept [37]-[39]. The frame of L slots is divided into two sections. C slots are reserved for circuit traffic and K = (L-C) slots are reserved for packet traffic. If no crossover is allowed then it is known as fixed boundary strategy. A scheme in which packet traffic is allowed to use any unused circuit slots is known as movable boundary strategy [40]. The modeling and performance analysis of such movable boundary hybrid switching schemes offer several complexities due to the extensive interactions between the circuit switched and the packet switched subsystems. Various methods have been used for the analysis [37]-[45].

(b) Switching strategy: The switching is done by an NxN nonblocking switch. Since each request for a connection is assumed to arise randomly, and the switch may not send two packets to the same output during the same slot, some scheduling must be done to avoid conflict at the switch output. Two types of scheduling are possible;

(I) Optimum Scheduling - Rearrangement of the existing calls are possible to accommodate a new call provided it meet other criteria.

(II) Random Scheduling - A call can be granted only if there is at least one pair of common time slots at the input and at the output.

However the much computationally difficult optimal scheduling would only increase the offered load by 10 to 15 percent, [35], compared to the much simpler random scheduling. We are only considering the random scheduling for the proposed circuit-packet switch.

Next Section presents the model description and chapter 7 presents the analysis. Chapter

8 provides the performance results and conclusion.

5.2 PROBLEM DESCRIPTION

The system under consideration is shown in Figure 5.1. N independent sources are connected to separate ports of an NxN nonblocking switch. Each user is allowed L time slots per frame out of which C time slots per frame is to transmit circuit traffic and K = (L-C) time slots for packet traffic. Any unused circuit time slots may also be used by packet traffic but preemptive priority of circuit traffic over packet is enforced. Thus a packet using a circuit slot is immediately put back into the input buffer when a circuit call arrives. Each time slot corresponds to one circuit connection or one packet connection. The switch distributes these packets to the appropriate sinks. If a circuit call arrives at input i for output j for which no circuit time slot is available then this call is blocked. If a packet arrives at input i for output j for which no time slot is available then this packet is queued behind the existing input queue to be served in the FCFS order when a time slot becomes available.

We assume infinite buffer for our analysis in the next section. We assume that each source request for a circuit connection is Poisson with rate λ_c and the probability that any given call is destined for any given output is 1/N. The call holding time is exponentially distributed with mean $1/\mu_c$ and circuit utilization is $\rho_c = \lambda_c/\mu_c$. Any new circuit arrival which can not be routed within a frame size will be rejected. We also assume that each source request for a packet connection is Poisson with rate λ_p and the probability that any given packet is destined for any given output is 1/N. The packet holding time

is exponentially distributed with mean $1/\mu_p$ and the packet slot utilization is $\rho_p = \lambda_p/\mu_p$. Also we assume $\alpha = \frac{\mu_p}{\mu_c} = \frac{1/\mu_c}{1/\mu_p} \geq 10^4$ as is the usual case for voice and data [46]. The newly arrived packets will be queued at the end of the input queue. We assume that the system operates in underload region. i.e., $\rho_p < k$. In this case the movable boundary strategy is used in order to reduce the packet delay time by utilizing circuit time slots whenever available. If we try to operate in the overload region to increase the throughput by utilizing the unused circuit slots, extraordinary long delays may result [42].

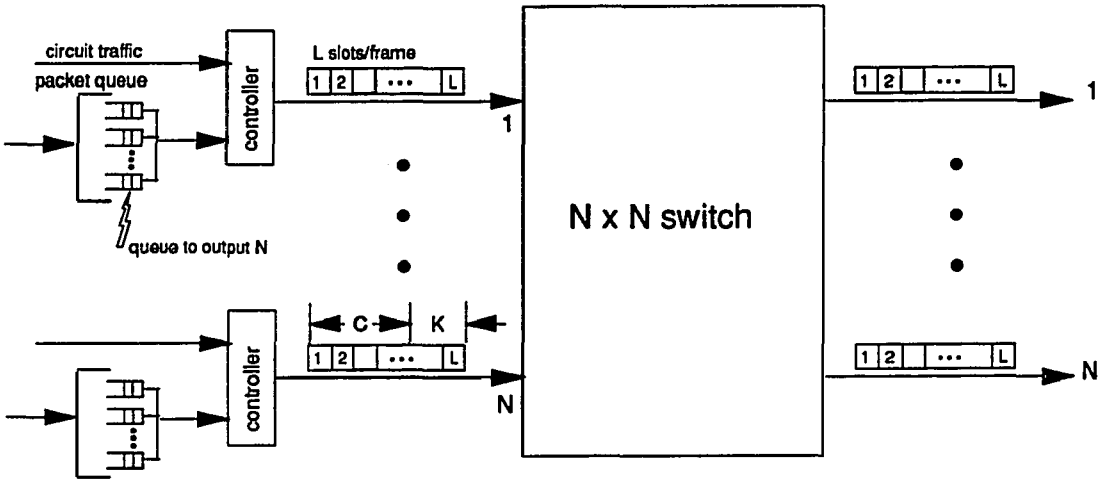


Fig. 5.1. Conceptual diagram of the proposed switch

During each frame the ongoing circuit traffic is defined as $T = [t_{ij}]$ and the ongoing packet traffic is defined as $D = [d_{ij}]$ where t_{ij} and d_{ij} are the number of on going circuits and packets (time slots) from source i to sink j. Both t_{ij} and d_{ij} are non negative integers. When a request for a new circuit connection comes it will be granted provided it meets the following scheduling criteria within a frame period.

(1) The total number of circuits in any input/output should not exceed C in a frame.

(2) There must be a common time slot (time slot coincident at time) available both at the input and at the output frame.

The unused circuit slots and the reserved packet slots (L-C) are used for the packet traffic if it meets the following criteria.

(1) The total number of slots (circuits + packets) in any input/output should not exceed L in a frame.

(2) There must be a common time slot available both at the input and at the output frame.

An illustrative example is shown in Figure 5.2 for N = 3, L = 5, C = 3 and K = 2. During time slot (TS) #1, input 2 is connected to output 2'. During TS #2, input 2 is connected to output 1' and input 3 is connected to output 2'. Input and output frames for each input and output are also shown. Now, let a new request arrive at input 1 for output 2'. There is a common time slot between input 1 and output 2'. If the new request is a circuit call it will be blocked since there already 3 ongoing circuit calls at output 2'.(note that the maximum number of circuit calls at any input or output (C) = 3). If it is a packet call it will be granted provided there were no other packets in queue to go from input 1 to output 2'.

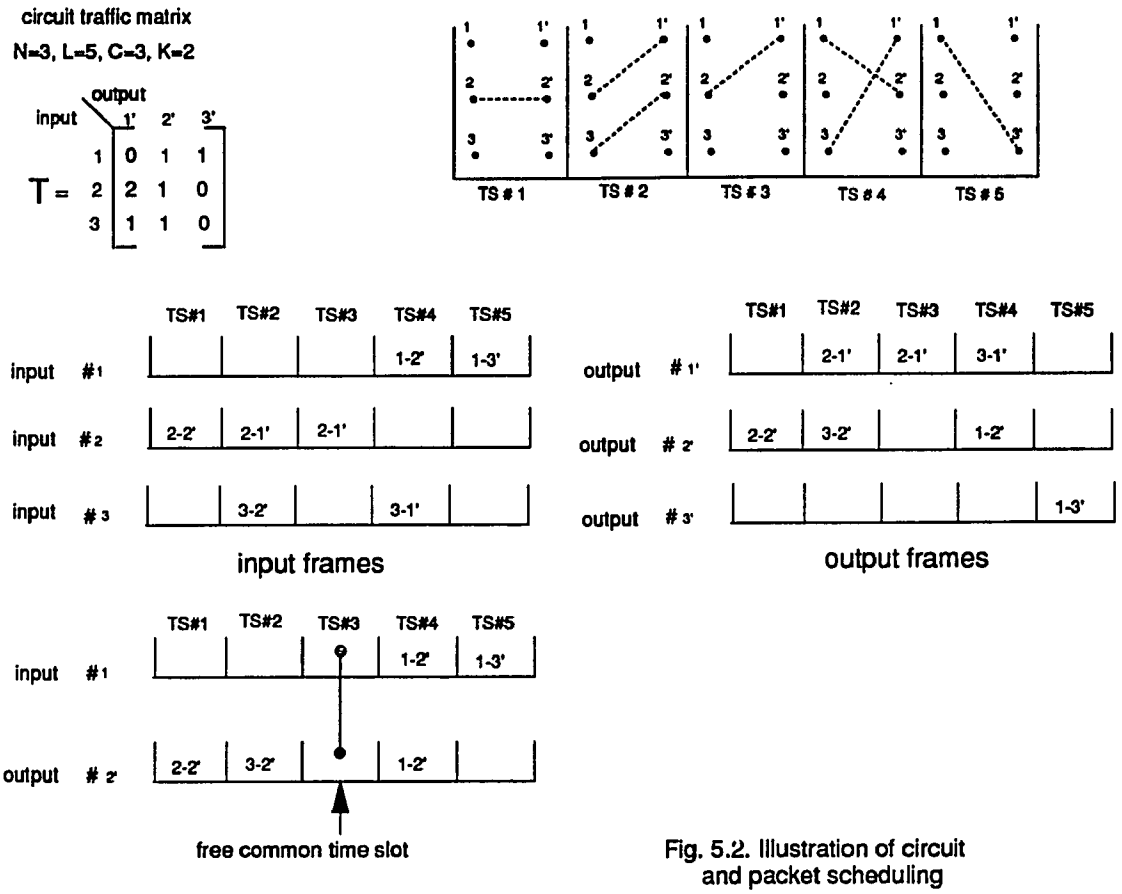


Fig. 5.2. Illustration of circuit and packet scheduling

In the next chapter we provide the analysis of the proposed scheme. The idea of the analysis is based on obtaining the Markov chain that describes the state of each input and output port.

6 ANALYSIS

A two dimensional Markov chain is used for the analysis. The state of the Markov chain (i,j) on a given input or output port is the following. 'i' is the number of active circuit calls, 'j' is the number of packets (in service + in queue). For an output port f, 'i' is the number of ongoing circuit calls, 'j' is the number of packets to a particular destination output f, either under service or waiting in input buffers for service. Let p(i,j) = Prob[i active circuit calls, j packets in the system]. When the number of ports N is large enough each I/O port chain approach dynamic independence of any other I/O chain [35]. A node of the Markov chain is shown in Figure 6.1. The transition rates from node (i,j) are the following, see Figure 6.1.

$\lambda_c(i,j)$: circuit arrival rate from the current state (i,j)

$\lambda_p(i,j)$: packet arrival rate from the current state (i,j)

$\mu_c(i,j)$: circuit departure rate from the current state (i,j)

$\mu_p(i,j)$: packet departure rate from the current state (i,j)

Evaluation of transition rates:

(a) Evaluation of $\lambda_c(i,j)$ and $\mu_c(i,j)$:

In the movable boundary strategy, circuit calls are not affected by packets because circuit calls have preemptive priority over packets. Circuit blocking probability P_B and the transition rates $\lambda_c(i,j)$, $\mu_c(i,j)$ can be obtained by decoupling circuit analysis from packets, see

Figure 6.2.

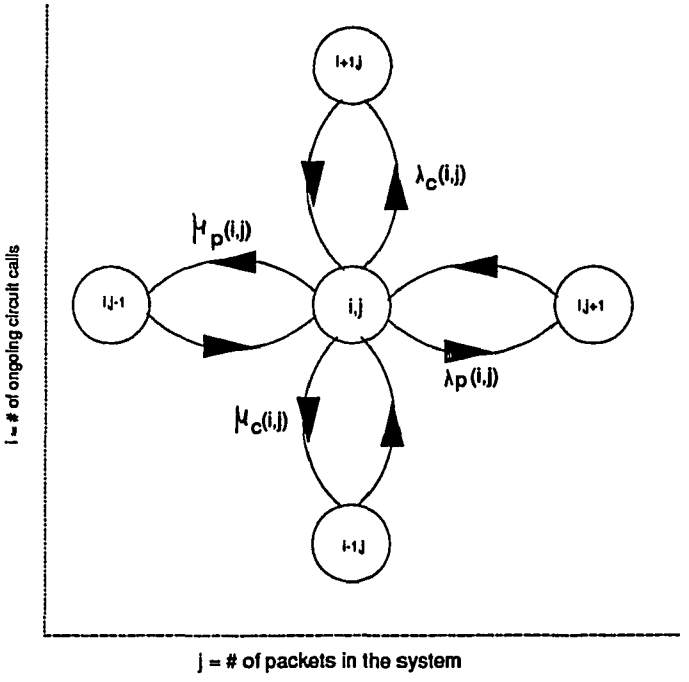


Fig. 6.1. Node (i,j) of the two dimensional Markov chain

for $1 \leq i \leq c-1$, $1 \leq j < \infty$

where i = # of active circuit calls, j = # of packets in the system

Then, the call acceptance rates are:

$$\lambda_c(i,0) = \lambda_c(i,1) = \lambda_c(i,2) = \ldots = \lambda_c(i,j) \equiv \lambda_c(i) \qquad (6.1)$$

Assume i < C circuit calls are active in the port under consideration. As mentioned earlier the circuit arrival rate is λ_c. The newly arrived call will not be accepted if the output port under consideration has already C ongoing calls or they do not have a common time slot

available. Thus define λ_i as

$$\lambda_i = \lambda(1 - Pr[\textit{output time slot unavailable/i circuit calls active}]) \qquad (6.2)$$

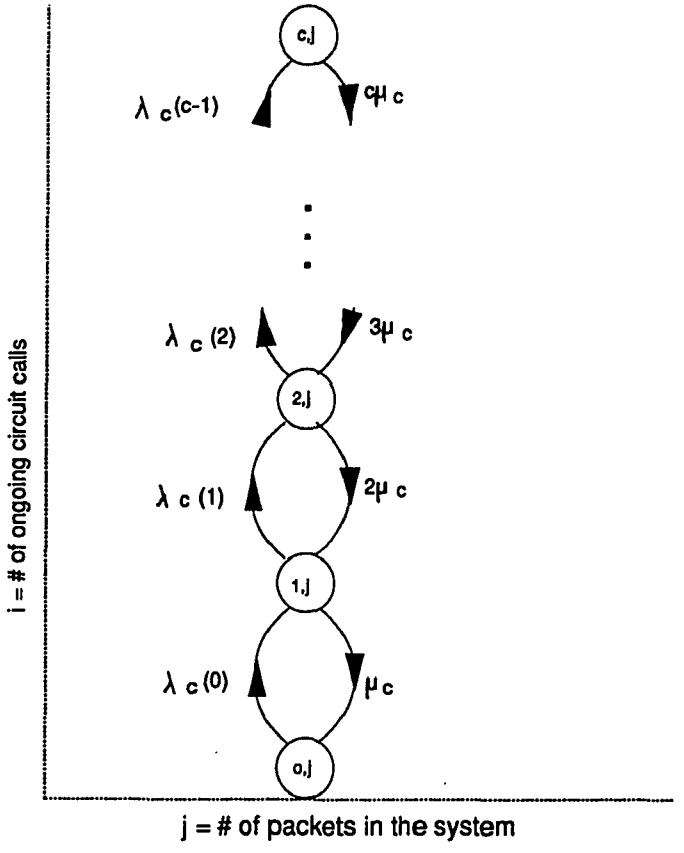


Fig. 6.2. Markov chain for circuit traffic

Let i and σ be the number of circuit calls at the input and the output respectively and $P(\sigma)$ be the probability that there are σ ongoing calls at the output. Then,

$$Pr[blocked/i] = \sum_{\sigma=0}^{C} P(\sigma) Pr[blocked/\sigma, i] \tag{6.3}$$

If $i+\sigma < L$ then a common free time slot must exist since the σ occupied slots could not span the (L-i) free input slots. For $i+\sigma \geq L$ blocking occurs only if (L-i) of the σ occupied output time slots coincide with the (L-i) unoccupied input time slots. Thus out of $\binom{L}{\sigma}$ possible time slot arrangements, $\binom{i}{L-\sigma}$ will result in blocking. Therefore,

$$Pr[blocked \ / \ i, \sigma] = \frac{\left[\begin{matrix} i \\ L-\sigma \end{matrix}\right]}{\left[\begin{matrix} L \\ \sigma \end{matrix}\right]} \quad for \quad i+\sigma \geq L \tag{6.4}$$

Using (6.2), (6.3) and (6.4) λ_i can be calculated as follows:

$$\lambda_i = \lambda_c \left(1 - \sum_{\sigma=L-i}^{C-1} P(\sigma) \frac{i!\sigma!}{L!(i+\sigma-L)!} - P(c) \right) \quad for \quad i > L-C \tag{6.5}$$

$$\lambda_i = \lambda_c (1 - P(c)) \quad for \quad i \leq L-C \tag{6.5-a}$$

The probability of blocking for a given C (reserved number of circuit slots), L (frame size)

and λ_c (circuit traffic arrival rate) is given by:

$$P_B = \sum_{k=0}^{C} P(k) Pr[blocked/k] \tag{6.6}$$

$$= \sum_{k=0}^{C} P(k) \left[\sum_{\sigma=L-k}^{C-1} P(\sigma) \frac{k!\sigma!}{L!(k+\sigma-L)!} + P(c) \right] \tag{6.6-a}$$

$$\mu_c(i,0) = \mu_c(i,1) = \ldots = \mu_c(i,j) \equiv \mu_c(i) = i\mu_c \tag{6.7}$$

See reference [35] for a more detailed analysis of circuit traffic.

(b) Evaluation of $\lambda_p(i,j)$:

$\lambda_p(i,j) = \lambda_p$ for all i and j since infinite buffer capacity is assumed at the input.

(c) Evaluation of $\mu_p(i,j)$:

Calculation of $\mu_p(i,j)$ is slightly more complex, therefore it is illustrated by a simple example with L = 2, C = 1, K =1 and we would continue with this example to find the improvement in queueing delay for the packet traffic due to movable boundary strategy compared to fixed boundary strategy.

To route a packet to a given output we have to know the number of ongoing circuit calls

and packet calls at that output port. To find this state probabilities we define a new two dimensional Markov chain with state (i,j) where i is the number of ongoing circuit calls and j is the number of ongoing packet calls. O(i,j) is the steady state probability that the system is in state (i,j). Figure 6.3 shows the Markov chain for L = 2, C = 1, K = 1.

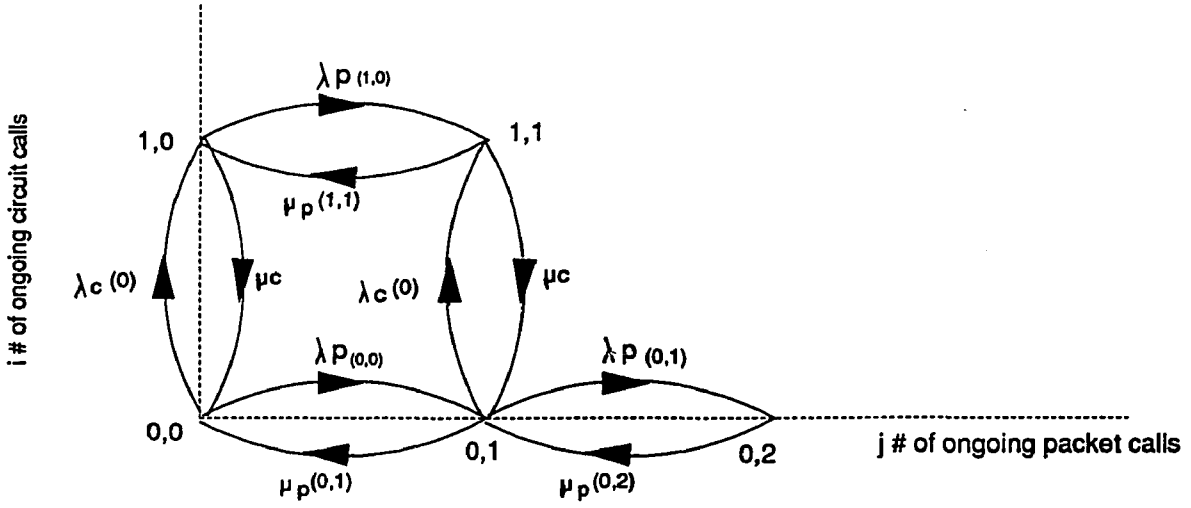


Fig. 6.3. Markov chain to calculate state probabilities

State O(i,j) where i = # of ongoing circuit calls
and j = # of ongoing packet calls

Assume the system is in state (0,0). The state can move to (0,1) if a packet arrives and is routed or there is already a packet in the queue and is routed. The packet can be routed if there is at least one free time slot at the requested output port. Packet is queued when the newly arriving packet can not be routed or a packet is preempted. Preempted packet arrives

with rate $O(0,1)\lambda_c(0)/2$. When the system is in state (0,1) and a circuit arrives with rate $\lambda_c(0)$, it can use any of the 2 slots. i.e., it can preempt the existing packet call with probability 1/2. Therefore,

$$\lambda_p(0,0) = [\lambda_p + rate\ \ packet\ \ is\ \ queued](1 - Pr[output\ \ slot\ \ unavailable]) \quad (6.8)$$

$$= [\lambda_p + \lambda_p(Pr[both\ \ the\ \ output\ \ slots\ \ are\ \ busy]) + arrival\ \ rate\ \ of\ \ preempted\ \ packet](1 - Pr[both\ \ the\ \ output\ \ slots\ \ are\ \ busy]) \quad (6.8-a)$$

$$= (\lambda_p + \lambda_p[O(1,1) + O(0,2)] + O(0,1)\lambda_c(0)1/2)(1 - [O(1,1) + O(0,2)]) \quad (6.8-b)$$

Pr[output slot unavailable] can be found from the case shown in Figure 6.4.

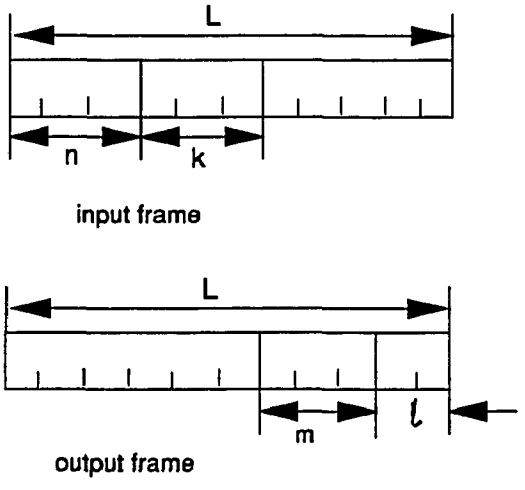


Fig. 6.4. Illustration of coincidence and noncoincidence of free time slots.

As shown in Figure 6.4 no blocking occurs when n+m+k+l < L.

$$Pr[time\ slot\ unavailable\ for\ packet/n,m,k,l] = \frac{\begin{bmatrix} n+k \\ L-(m+l) \end{bmatrix}}{\begin{bmatrix} L \\ (m+l) \end{bmatrix}} \tag{6.9}$$

$$= \frac{(n+k)!(m+l)!}{L!(n+k+m+l-L)!} \quad for \quad n+k+m+l \geq L \tag{6.9-a}$$

$$\lambda_p(0,1) = (\lambda_p + \lambda_p[O(1,1) + O(0,2) + O(1,0)/2 + O(0,1)/2] + O(0,2)\lambda_c(0))$$
$$(1 - [O(1,1) + O(0,2) + O(1,0)/2 + O(0,1)/2]) \tag{6.10}$$

$$\lambda_p(1,0) = (\lambda_p + \lambda_p[O(1,1) + O(0,2) + O(1,0)/2 + O(0,1)/2])$$
$$(1 - [O(1,1) + O(0,2) + O(1,0)/2 + O(0,1)/2]) \tag{6.11}$$

Assume the state is (0,1). The transition to state (0,0) can happen if the packet finish service or the packet is preempted.

$$\mu_p(0,1) = \mu_p + \lambda_c(0)/2 \tag{6.12}$$

$$\mu_p(0,2) = 2\mu_p + \lambda_c(0) \tag{6.13}$$

$$\mu_p(1,1) = \mu_p \tag{6.14}$$

$\lambda_c(0)$ And μ_c were found in equations (6.5) and (6.7). Solving the Markov chain we get the probability of different states. We can only solve the Markov chain numerically because the state equations are nonlinear.

Now consider the original Markov chain with state P(i,j), (see Figure 6.1) where i is the number of ongoing circuit calls, j is the number of packets (both in queue and in service). Figure 6.5 shows this Markov chain for L = 2, C = 1, K = 1.

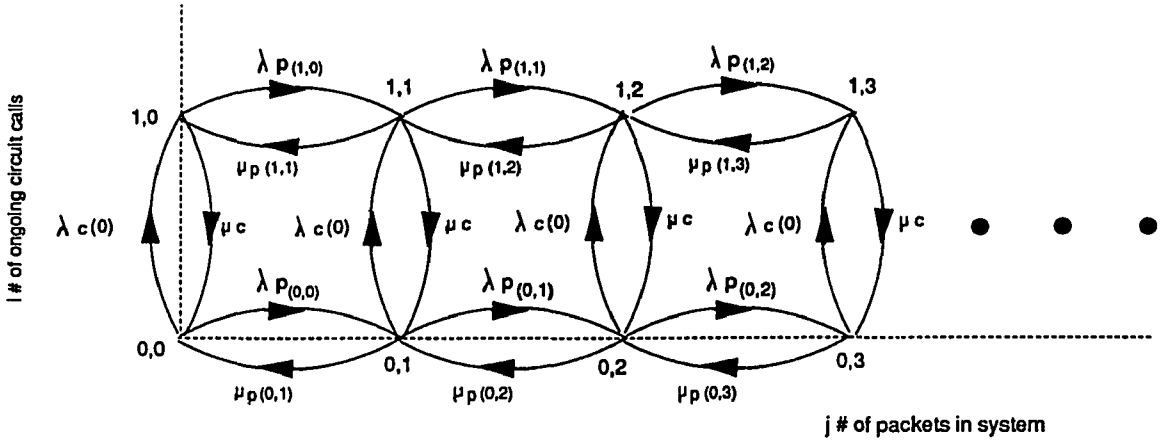


Fig. 6.5. Markov chain to calculate packet delay

State O(i,j) where i = # of ongoing circuit calls
and j = # of ongoing packet calls

Here we know all the transition probabilities except μ_p's. Assume the system is in state (1,0). The packet can be routed to the output if any of the slots in the requested output port is free. Therefore,

$$\mu_p(0,1) = \mu_p(1 - Pr[output \quad slot \quad unavailable]) \tag{6.15}$$

$$= \mu_p(1 - [O(1,1) + O(0,2)]) \quad = \quad b\mu_p \tag{6.15-b}$$

where b = 1 - [O(1,1) + O(0,2)].

$$\mu_p(1,1) = \mu_p(1 - [O(1,1) + O(0,2) + O(1,0)/2 + O(0,1)/2]) \quad = \quad a\mu_p \tag{6.16}$$

where a = $1 - [O(1,1) + O(0,2) + O(1,0)/2 + O(0,1)/2]$

$$\mu_p(0,2) = \mu_p(0,1) + \mu_p(1 - [O(1,1) + O(0,2) + O(1,0)/2 + O(0,1)/2]) \quad = \quad c\mu_p \tag{6.17}$$

where c = a + b.

Five balance equation can be written by inspection.

$$(a\mu_p + \mu_c + \lambda_p)p_{1j} = \lambda_{c(0)}p_{0j} + a\mu_p p_{1j+1} + \lambda_p p_{1j-1} \quad j \geq 1 \tag{6.18}$$

$$(\lambda_{c(0)} + \lambda_p + c\mu_p)p_{0j} = c\mu_p p_{0j+1} + \mu_c p_{1j} + \lambda_p p_{0j-1} \quad j \geq 2 \tag{6.19}$$

$$(\lambda_p + \mu_c)p_{10} = \lambda_{c(0)}p_{00} + a\mu_p p_{11} \tag{6.20}$$

$$(\lambda_{c(0)}+\lambda_p)p_{00}=b\mu_p p_{01}+\mu_c p_{10} \tag{6.21}$$

$$(\lambda_p+\lambda_{c(0)}+b\mu_p)p_{01}=c\mu_p p_{02}+\mu_c p_{11}+\lambda_p p_{00} \tag{6.22}$$

Define

$$G_i(z)\equiv\sum_{j=0}^{\infty}p_{ij}z^j \quad i=0,1 \tag{6.23}$$

Multiplying equations (6.18) and (6.19) by z^j and summing over all values of $j \geq 1$ and $j \geq 2$ respectively, we get two algebraic equations that involve the two generating functions $G_0(z)$ and $G_1(z)$. Solving for $G_0(z)$ and $G_1(z)$ we get,

$$G_0(z)=\frac{z(cp_{00}+ap_{01}z+ap_{10})+\alpha(z-1)(a-\rho_p z)(cp_{00}+ap_{01}z)}{\alpha(c-\rho_p z)(z-1)(a-\rho_p z)+z(c-\rho_p z)+\rho_c z(a-\rho_p z)} \tag{6.24}$$

$$G_1(z)=\frac{\rho_c z G_0(z)+\alpha a(z-1)p_{10}}{\alpha(z-1)(a-\rho_p z)+z} \tag{6.25}$$

The average number of packets in the system is given by:

$$E(j)=\sum_{j=0}^{\infty}j(p_{0j}+p_{1j}) \quad = \quad G_0'(1)+G_1'(1) \tag{6.26}$$

where $G_i'(1)=\left.\frac{dG_i(z)}{dz}\right|_{z=1}$ $\quad for i=0,1$

Using Little's theorem we can find the normalized total delay:

$$\mu_p E(T) = E(j)/\rho_p = \frac{a^2 + a + c + \rho_c[c + (a/b)\rho_p]}{[b + (a - \rho_p)(1 + \rho_c)]\,[c + (a/b)\rho_p]} \tag{6.27}$$

The normalized wait time is given by:

$$\mu_p E(W) = \mu_p E(T) - 1 \tag{6.28}$$

In order to compare the performance of movable boundary strategy with that of fixed boundary strategy, below we provide the results for fixed boundary strategy.

Determination of $\mu_p E(W)$ for fixed boundary

Figure 6.6 shows the Markov chain with arrival rate λ_p and departure rate $\mu_p(1 - p_1)$. The state of the chain is number of packets in the system. The service rate is reduced by the factor (1 - p_1) because a packet call can be scheduled only if the output slot is free. To find p_1 we construct another Markov chain with state as the number of packets in service, see Figure 6.7. We define the packet slot utilization for the fixed boundary case, $\rho_{pf} = \lambda_p/[\mu_p(1 - p_1)]$. Therefore,

$$\mu_{pf} E(W) = \rho_{pf}/(1 - \rho_{pf}) \tag{6.29}$$

where μ_{pf} is the packet service rate for the fixed boundary case.

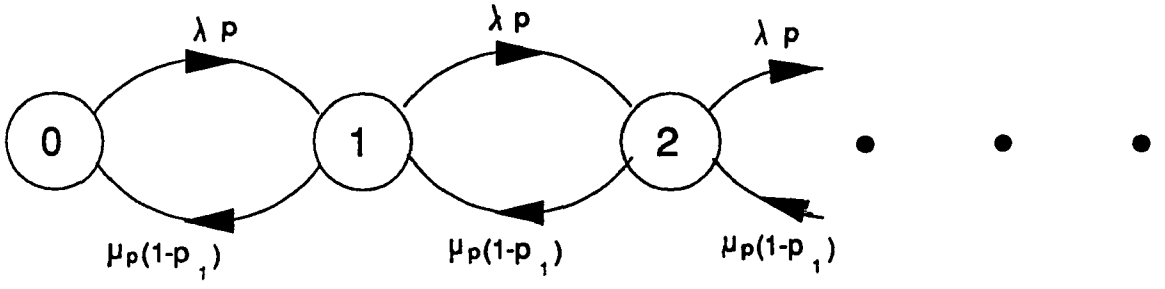


Fig. 6.6. Markov chain for fixed boundary packet traffic with K = 1 and the state as the # of packets in system (in service + in queue)

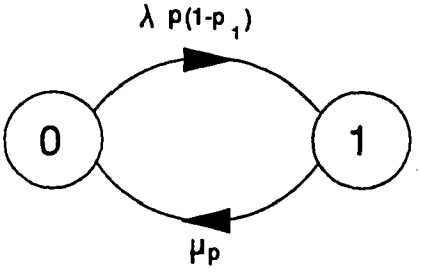


Fig. 6.7. Markov chain for fixed boundary packet traffic with K = 1 and state as the # of packets at service

7 RESULTS AND CONCLUSION

Figure 7.1 shows packet offered load ρ_p versus the normalized wait time for various values of the circuit offered load ρ_c. The performance of the movable boundary strategy is better than that of the fixed boundary strategy. For example in the fixed boundary strategy the wait time is 2 packet service time when the utilization is 0.48. In the movable boundary strategy the utilization is 0.71 ($\rho_c = 0.8$) or even 0.97 ($\rho_c = 0.1$) for the same wait time of 2. This improvement is due to the fact that the unused circuit slots are used by packet traffic. Circuit blocking probability is also improved when integrated with packet traffic. This is obvious by comparing the two curves in Figure 7.2 (a) and (b). For example when the circuit load (ρ_c) = 0.6 and C = 7, Figure 7.2 (b), the circuit blocking probability (P_B) is 2.2×10^{-5} for circuit switching only and P_B is 6.1×10^{-6} for integrated switching. This improvement is caused by the fact that the frame size increased from C to L and thus blocking is reduced during random scheduling.

The proposed switch can be used to handle normal circuit traffic and interactive data as packet traffic. The proposed switch can take advantage of the interactive user *think time* [47] and thus results in better utilization of the transmission and switching facilities.

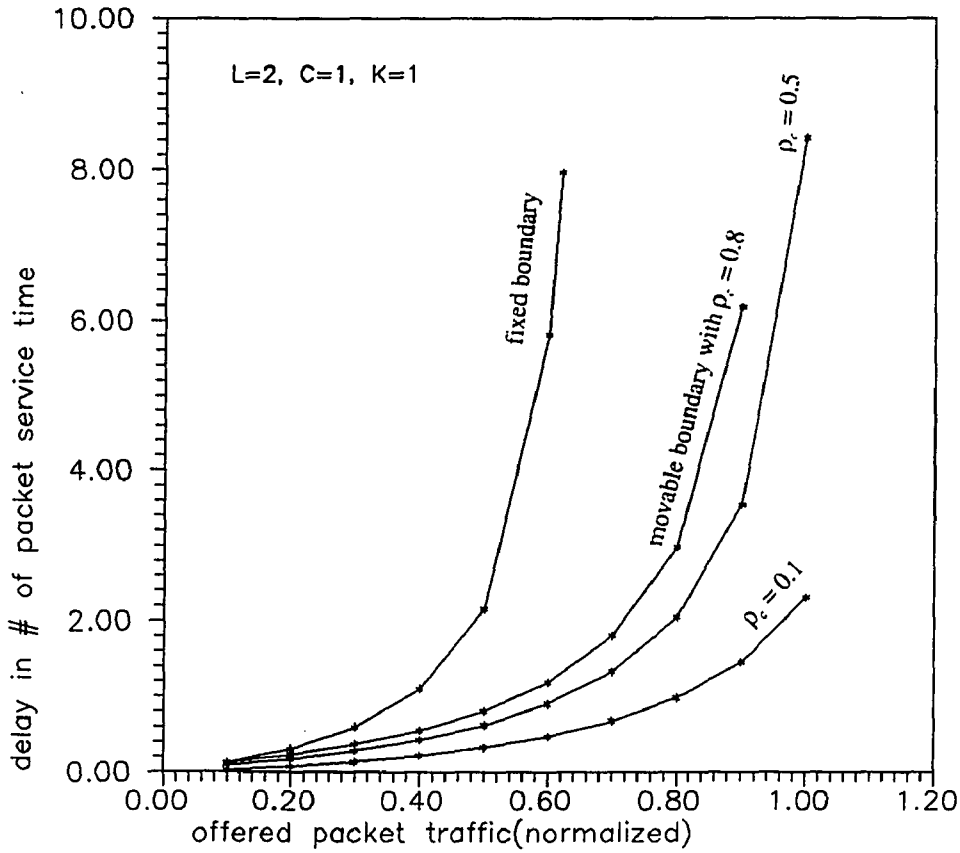


Fig. 7.1. **Comparison of queueing delay in fixed boundary and in movable boundary**

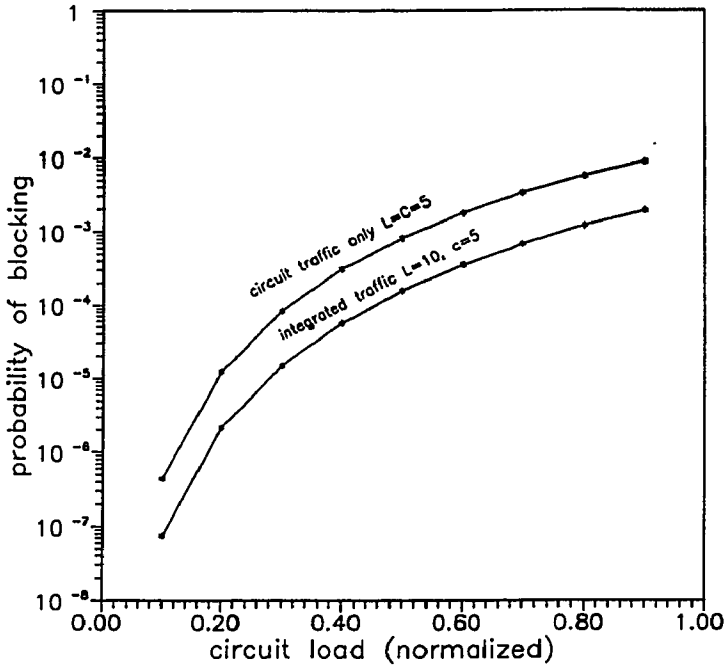


Fig. 7.2a. Comparison of blocking probability in integrated switching and in circuit switching (L=10, C=5)

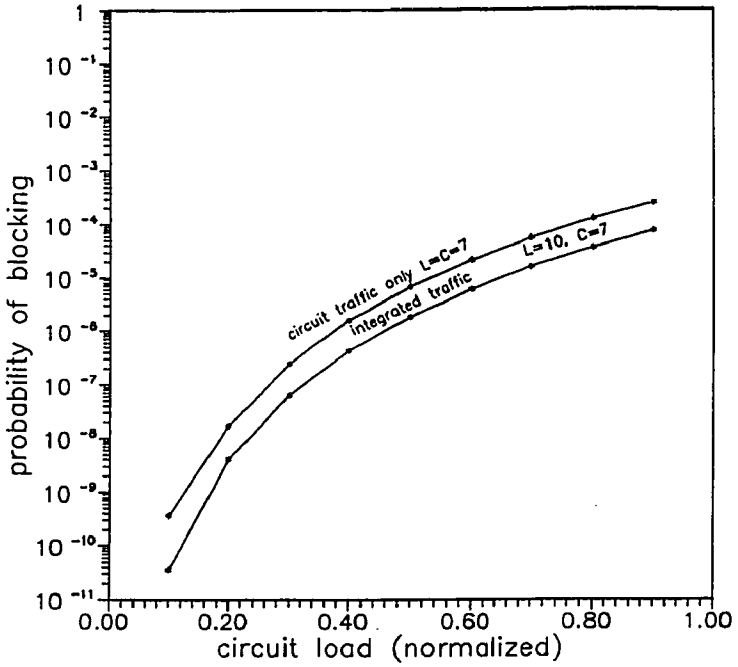


Fig. 7.2b. Comparison of blocking probability in integrated switching and in circuit switching (L=10, C=7)

8 FUTURE WORK

The proposed switch in chapter 2 uses a variable length frame for switching. The frame length can vary from 0 to M. It is interesting to see how the queueing delay and buffer overflow probability varies if we use a fixed length frame for switching instead of the variable length frame. We can find an algorithm to schedule L number of packets if the fixed frame size is L when the number of packets waiting in input buffers are greater than or equal to L packets). Once we have an algorithm we can find the queueing delay for each packet. In steady state the arrival of packets during a frame length will be equal to the sum of the Poisson arrival of fresh packets and the unscheduled packets already in the buffers.

The queueing delay analysis of the integrated circuit-packet switch has been carried out only for a special case in chapter 7. Eventhough a general solution is practically impossible, analysis for more special cases can be done. Another interesting problem would be to find the optimum boundary for a given circuit and packet load. In other words we should be able to move the boundary according to the traffic conditions and priority. For example, if we want a particular probability of blocking for circuit traffic then we should be able to find the optimum boundary. The boundary may have to move for changing load conditions to keep the probability of blocking acceptable or constant.

9 APPENDICES

Appendix I provides a review of some existing scheduling algorithms for demand assignment and Appendix II provides a list of symbols used in the analysis in chapter 3.

9.1 APPENDIX I

Scheduling Algorithms

Scheduling algorithm is the process of decomposing the traffic matrix T, into a number of switching matrices T_s, $1 \leq s \leq S$, where S is the total number of switching matrices. T_s has at most one nonzero entry in each row or column. The matrix T may be considered as the *adjacency matrix* of a weighted *bipartite graph* G_T, and matrix T_s of a *matched bipartite graph* (weighted) G_s. Figure 2.2b illustrates that any G_s configuration of the switch routes the traffic from input to output without conflict. The duration of a switching configuration is equal to largest entry in T_s, denoted by $|T_s|$. Hence the transmission length (L) can be written as

$$L = \sum_{s=1}^{S} |T_s| \qquad (I-1)$$

In order to describe the efficiency of a scheduling algorithm the following four parameters have been used:

1. Transmission length (L) (Defined by equation. I-1)
2. The service delay time (D_s), i.e., the elapsed time from the beginning of a TP until packet departure. (See equation. I-2 below).
3. The number of switching matrices (S)
4. The computational complexity of the algorithm.

Several algorithms have been presented in the literature [30]-[33], that optimize one parameter or the other. In [30] T. Inukai has developed a basic algorithm which minimizes the transmission length L, i.e., the algorithm always generates transmission length L=Z, where Z is the maximum line sum as defined by equation 2.2. In the description of this algorithm, which is given below, the following terms have been used.
Line, a row or column in T; *Critical line* any line with maximum line sum. A switching matrix *covers* a line if it has a nonzero entry on that line.

The algorithm

STEP 1: set 1 ← s and T(0) ← T

STEP 2: Find a scheduling matrix T_s which covers all critical lines of T(s) and with maximum number of nonzero entries.

Set T(s) ← T(s-1) - T_s. If T(s) equals zero stop, otherwise, repeat step 2.

Step 2 is achieved with the aid of a bipartite matching algorithm which finds a bipartite graph G_s of maximum cardinality at each iteration of step 2. T_s is the adjacency of the

graph G_s. An optimum transmission schedule is shown in the example of Figure 2.2c. The number of switching matrices S generated by the above algorithm is bounded by N^2-2N+2, while its computational complexity is polynomial, $O(SN^2)$.

A number of suboptimal algorithms have also been proposed which generate average transmission length ($\overline{L} = 1.2\overline{Z}$), but with reduced complexity. I. Gopal et al. in reference [31], examines the problem of minimizing the service time delay D_s. In defining D_s, each matrix T_s has been expressed as a sum of *permutation matrices u(l)*, (*u(l)* is a 0,1, matrix with at most one nonzero entry in each line). Then $T_s = \sum_{x=1}^{|T_s|} u(s+x-1)$ for each T_s and $1 \le s \le S$. This is equivalent to the sequence $u(l)$, $1 \le l \le L$. Let $[u(l)]$ be the number of 1's in $u(l)$, then D_s is defined by:

$$D_s = \frac{1}{t_0} \sum_{l=0}^{L} l[u(l)] \qquad (I-2)$$

where t_0 is the total traffic in T.

Reference [31] provides the lower bound on D_s. The proposed algorithm however is computationally explosive. It has also been shown that the necessary condition to achieve maximum D_s is to have minimum transmission length L = Z.

In minimizing the number of switching S ($S \le N$), the transmission length cannot be kept minimum (L>Z). In reference [32], we have succeeded in minimizing the number of switching configurations ($S \le N$) while keeping the transmission length L the smallest possible.

9.2 APPENDIX II

List of symbols

N:	Number of input or output ports
$T = [t_{ij}]$:	Traffic matrix with entries t_{ij}
r_i:	Row sum.
c_j:	Column sum.
R:	Maximum row sum.
C:	Maximum column sum.
Z:	Maximum line sum.
t_0:	Total traffic in T.
T_s:	Switching matrix.
$\|T_s\|$:	Switching duration.
$u(l)$:	Permutation matrix.
L:	Transmission length.
S:	Total number of switching matrices.
σ:	Probability of a packet arrival in a time slot, to any input i.
P_L:	Probability of having transmission length L; $\mathbf{P_L}$: the vector.
$H=[h_{nm}]$:	Matrix of transition probabilities h_{nm}.
P_B:	Buffer overflow probability.
γ:	Throughput.
M:	Input buffer capacity.

10 REFERENCES

[1] J. S. Turner and L. F. Wyatt, "A packet network architecture for integrated services," *in Proc. GLOBECOM'83*, San Diego, CA, Nov. 1983, pp. 2.1.1-2.1.6.

[2] J. J. Kulzer and W. A. Montgomery, "Statistical switching architecture for future services," *in Proc. ISS'84*, Florence, Italy, May 1984, pp. 43A.1.1-43A.1.6.

[3] D. M. Dias and J. R. Jump, "Packet switching interconnection networks for modular systems," *IEEE Comput. Mag.*, vol. 14, pp. 43-53, Dec. 1981.

[4] D. M. Dias and M. Kumar, "Packet switching in N log N multistage networks," *in Proc. GLOBECOME'84*, Atlanta, GA, Dec. 1984, pp. 114-120.

[5] Yih-Chyun Jenq, "Performance analysis of a packet switch based on single-buffered Banyan network," *IEEE J. Select. Areas Commun.*, Vol. SAC-1, Dec. 1983, pp. 1014-1021.

[6] J. Y. Hui and E. Arthurs, "A broadband packet switch for integrated transport," *IEEE J. Select. Areas Commun.*, vol. SAC-5, Oct. 1987, pp. 1264-1273.

[7] J. Y. Hui, "A broad packet switch for multi-rate services," *in Proc. ICC'87*, Seattle, WA, June 1987, pp. 782-788.

[8] Y. S. Yeh, M. G. Hluchyj, and A. S. Acampora, "The Knockout switch: A simple, modular architecture for high-performance packet switching," *IEEE J. Select. Areas Commun.*, vol. SAC-5, pp. 1274-1283, Oct. 1987.

[9] K. Y. Eng. M. G. Hluchyj, and Y. S. Yeh, "A Knockout switch for variable-length packets," *IEEE J. Select. Areas Commun.*, vol. SAC-5, pp. 1426-1435, Dec. 1987.

[10] H. Yoon, M. T. Liu, and K. Y. Lee, "The Knockout switch under nonuniform traffic," *in Proc. GLOBECOM '88*, Hollywood, FL, Nov. 1988, pp. 1628-1634.

[11] Satosh Nojoima et. al., "Integrated Services Packet Network using bus matrix switch," *IEEE J. Select. Areas Commun.*, vol. SAC-5, Oct. 1987, pp. 1284-92.

[12] M. De Prycker and M. De Somer, "Performance of a service independent switching network with distributed control," *IEEE J. Select. Areas Commun.*, vol. SAC-5, pp. 1293-1301, Oct. 1987.

[13] M. Karol, M. Hluchyj and S. Morgan, "Input versus output queueing on a space-division packet switch," *IEEE Trans. Commun.*, vol. COM-35, Dec. 1987, pp. 1347-56.

[14] M. G. Hluchyj and M. Karol, "Queueing in space-division packet switching," *in Proc. INFOCOM '88*, New Orleans, LA, Mar. 1988, pp. 334-343.

[15] M. Hluchyj & M. Karol, "Queueing in high-performance packet switching," *IEEE J. Select. Areas in Commun.*, Dec. 1988, pp. 334-343.

[16] J. S. Turner, "New directions in communications (or which way to the information age?)," *IEEE Commun. Mag.*, vol. 24, pp. 8-15, Oct. 1986. Also, *in Proc. Int. Zurich Seminar Digital Commun.*, Zurich, Switzerland, Mar. 1986, pp. A3.1-A3.8.

[17] J. D. Morse and S. J. Kopec Jr., "Performance evaluation of a distributed burst-switching communications system," *in Proc. Phoenix Conf. Comput. Commun.*, Mar. 1983, pp. 41-46.

[18] S. R. Amstutz, "Burst-switching-A method for dispersed and integrated voice and data switching," *IEEE Commun. Mag.*, vol. 21, pp. 36-42, Nov. 1983. Also, in Proc. ICC '83, June 1983, pp. 288-292.

[19] E. F. Haselton, "A PCM Frame switching concept leading to burst switching network architecture," *IEEE Commun. Mag.*, vol. 21, pp. 13-19, Sept. 1983. Also, in Proc. ICC'83, Boston, MA, June 1983, pp. 1401-1406.

[20] C. P. Kruskal and M. Snir, "The performance of multistage interconnection networks for multiprocessors," *IEEE Trans. Comput.*, vol. C-32, pp. 1091-1098, Dec. 1983.

[21] M. Kumar and J. R. Jump, "Performance of unbuffered shuffle-exchange networks," *IEEE Trans. Comput.*, vol. C-35, pp. 573-577, June 1986.

[22] K. E. Batcher , "Sorting networks and their application," *in Proc. Spring Joint Comput. Conf., AFIPS,* 1968, pp. 307-314.

[23] H. Ahmadi and W. E. Denzel, "A survey of modern high-performance switching techniques," *IEEE J. Select. Areas Commun.,* vol. SAC-7, pp. 1091-1103, Sept. 1989.

[24] H. Ahmadi et al., "A high-performance switch fabric for integrated circuit and packet switching," *in Proc. INFOCOM '88,* New Orleans, LA, Mar. 1988, pp. 9-18.

[25] A. Huang and S. Knauer, "Starlite: A wideband digital switch," *in Proc. GLOBECOM '84*, Atlanta, GA, Dec. 1984, pp. 121-125.

[26] J. S. Turner, "Design of a broadcast packet switching network," *in Proc. INFOCOM '86*, Apr. 1986, pp. 667-675.

[27] I. S. Gopal, I. Cidon, and H. Melis, " PARIS: An approach to integrated private networks," *in Proc. ICC '87*, Seattle, WA, June 1987, pp. 764-773.

[28] I. Cidon, I. S. Gopal, and S. Kutten, "New models and algorithms for future networks," *in Proc. ACM Principles Distributed Comput.,* Toronto, Canada, 1988.

[29] H. S. Kim and A. Leon-Garcia, "A self-routing multistage switching network for broadband ISDN," *IEEE J. Select. Areas Commun.*, vol. SAC-8, pp. 459-466, Apr. 1990.

[30] T. Inukai, "An efficient SS/TDMA time slot assignment algorithm," *IEEE Trans. Commun.,* vol. COM-27, Oct. 79, pp. 1449-55.

[31] I. Gopal, D. Coppersmith and C. K. Wong, "Minimizing packet waiting time in a multibeam satellite system," *IEEE Trans. Commun.*, COM-30, Feb. 1982, pp. 305-316.

[32] D. P. Gerakoulis, T. N. Saadawi and D. L. Schilling, "Minimizing schedule length in a SS/TDMA system with minimum number of switching," *Proceedings of IEEE GLOBECOME* 1985.

[33] R. A. Spanke, "Architectures for guided-wave optical space switching systems," *IEEE Communications magazine*, May 1987, pp. 42-48.

[34] S. M. Barta and M. L. Holig, "Analysis of a demand assignment TDMA blocking system," *AT&T Bell Labs Technical Journal* vol. 63, No. 1, Jan. 1984, pp. 89-114.

[35] C. Rose and M. Hluchyj, "The performance of random and optimal scheduling in a time-multiplex switch," *IEEE Trans. Commun.*, COM-35, Aug 1987, pp. 813-817.

[36] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, New York: McGraw-Hill, 1984.

[37] G. J. Coviello and P. A. Vena, "Integration of Circuit/Packet Switching by a SENET (Slotted Envelope Network) Concept," *Proceedings of the National Telecommunications Conference*, pp. 42-12-42-17, Dec. 1975.

[38] A. Leon-Garcia, R. Kwong and G. F. Williams, "Performance Evaluation Methods for an Integrated Voice/Data Link," *IEEE Trans. Commun.*, vol. COM-30, no. 8, pp. 1848-1858, Aug. 1982.

[39] D. P. Gaver and J. H. Lehoczky, "Channels that Cooperatively Service a Data Stream and Voice Messages," *IEEE Trans. Commun.*, vol. COM-30, no. 5, pp. 1153-1161, May 1982.

[40] K. Kummerle, "Multiplexer Performance for Integrated Line and Packet Switched Traffic," *Proceedings of the International Conference on Computer Communications*, pp. 507-515, Aug. 1974.

[41] M. J. Fisher and T. C. Harris, "A Model for Evaluating the Performance of an Integrated Circuit- and Packet-Switched Multiplex Structure," *IEEE Trans. Commun.*, vol. COM-24, no. 2, pp. 195-202, Feb. 1976.

[42] C. J. Weinstein, M. L. Malpass and M. J. Fisher, "Data Traffic Performance of an Integrated Circuit- and Packet-Switched Multiplex Structure," *IEEE Trans. Commun.*, vol. COM-28, no. 6, pp. 873-878, June 1980.

[43] K. Sriram, P. K. Varshney and J. G. Shanthikumar, "Discrete- Time Analysis of Integrated Voice/Data Multiplexers with and without Speech Activity Detectors," *IEEE J. Select. Areas in Commun.*, vol. SAC-1, no. 6, pp. 1124-1132, Dec. 1983.

[44] S. Q. Li and J. W. Mark, "Performance of Voice/Data Integration on a TDM System," *IEEE Trans. Commun.*, vol. COM-33, no. 12, pp. 1265-1273, Dec. 1985.

[45] G. F. Williams and A. Leon-Garcia, "Performance Analysis of Integrated Voice and Data Hybrid-Switched Links," *IEEE Trans. Commun.*, vol. COM-32, no. 6, pp. 695-706, June 1984.

[46] M. Schwartz, *Telecommunication Networks*, Reading, MA: Addison-Wesley, 1987, ch. 12.

[47] E. Harrington, "Voice/data integration using circuit switched networks," *IEEE Trans. Commun.*, June 1980, pp. 781-793.