77-42

ANALYSIS OF INTEGRATED VOICE AND DATA COMMUNICATION NETWORK

Lih-Hsing Chang

Department of Electrical Engineering
Carnegie-Mellon University
Pittsburgh, Pa.15213

November 1977

This work was supported by Defense Comunication Agency under contract number (DCA100-76-0058).

ABSTRACT

This thesis studies the performance of an integrated voice and data communication network. Different characteristics from other communication networks and a different set of key parameters for its performance are addressed. The relationship of and the trade-off between the key parameters are also discussed.

Voice communication requires slow but continuous information exchange while data communication requires burst type of information exchange. A new integrated switch is designed to support both type of communications; line switch for voice and packet switch for data. Class I traffic, voice or video, is modeled as an M/M/n queue and Class II traffic, data or bulk, is modeled as an M/M/Y queue. A wild distribution of Class II queue length is discovered and a significant trade-off between communication and ccmputer facilities is implied. The study shows that the queue length grows very rapidly when the Class I/Class II job size ratio increases. A small integrated switch with relatively small job size ratio is studied in details. However large switches with realistcal job size ratio are only approximated and detail quantitative results for such system require further study.

The longer the queue is, the larger the memory will be required to store the data packets. However though the integrated switch has a very long queue, the number of buffer for the switch to operate efficiently is rather limited as shown by the network queueing model. In order to increase the memory utilization and to lower the system cost, several memory management and buffer assignment schemes are discussed. An unconventional secondary storage for switching processor is modeled and the advantages and disadvantages are thoroughly discussed.

A special delay of integrated switch is introduced by its frame structure. In order to minimize the through network delay, the design problem of how to share the communication link capacities between voice and data and how to assign the frame skew on each link are discussed. Because of discrete delay incremented by the frame period, the frame skew assignment problem is investigated as a mixed integer linear program. A speeding algorithm using k-tree concept is developed to speed up the ordinary branch and bound algorithm. However the speeding algorithm does not show much improvement in computation time and is thus limited useful. A heuristic algorithm is then developed to find a local optimal assignment in relatively short computation time.

ACKNOWLEDGEMENT

First and foremost my thanks go to my advisor, Professor Sam Fuller, for his enlightened supervision of this work, and to the other members of my committee, Professor Dan Sieworek, John Lehoczky and Dr. Mario Barbacci for reading innumerable version of the draft and for giving thoughtful comments.

I am further indebted to the DCA group whom I have worked with and received various comments and suggestion from for the past two years.

I like to give thanks to all my friends who gave me so many help and encouragement during the past several years. Special thanks go to Mr. Chen and Mr.Tung who host me when I come back to finish this work.

TABLE OF CONTENTS

ANALYSIS OF
INTEGRATED VOICE AND DATA COMMUNICATION NETWORK

CHAPTER 3 Memory Management for Data Buffers

CHAPTER 1 Introduction

1.1. Computer Communication

The computer communication field is some 20 years old now, having originated with military command and control system in which widely dispersed radar inputs and outputs to weapon sites had to be controlled rapidly and in real time by a central facility. That central facility could only be a digital computer because of the speed and data volume involved. Today, data-processing systems having a communications network as an integral "central nervous system" are a growing part of our daily lives. These data processing systems are evolving at a bewildering speed and in a variety of directions. Their evolution is made possible by new hardware and software developments, new user needs, and new transmission systems tailored to computer communication.

During much of the 1960s, the growth of computer networks was hampered by the lack of communication facilities well suited for data transmission. Because the existing telecommunication networks designed and operated by commerical carriers had evolved in a manner conducive to voice communications, they could not readily provide the switching functions needed for the overall cost effective utilization of transmission facilities for interactive data communications. As a result, we witnessed an emergence of special networks such as DATRAN, ARPANET, and others dedicated to data users. Today separate packet switching networks are used widely for data communications.

Although there appear to be valid justifications for the above trend, the practice of separating voice and data traffic should be continually examined. The search for

switching approaches that would allow more versatility with respect to answering all communication needs should be encouraged. In this thesis we will take a look at a single communication network which provides both data and voice traffic services through the use of a switching approach that operates somewhere between a full circuit-switched and a full packet-switched concept.

As shown in figure 1.1.1, a computer communication network usually consists of terminals, local loops, switching centers (SC), and high speed trunks, which form the backbone communication system. A terminal may be a teletype, a telephone or a computer. A message generated by a terminal is transmitted by local loops to a switching center. From the switching center, the message is carried by a high speed trunk, in exactly inverse order, to the destination terminal.

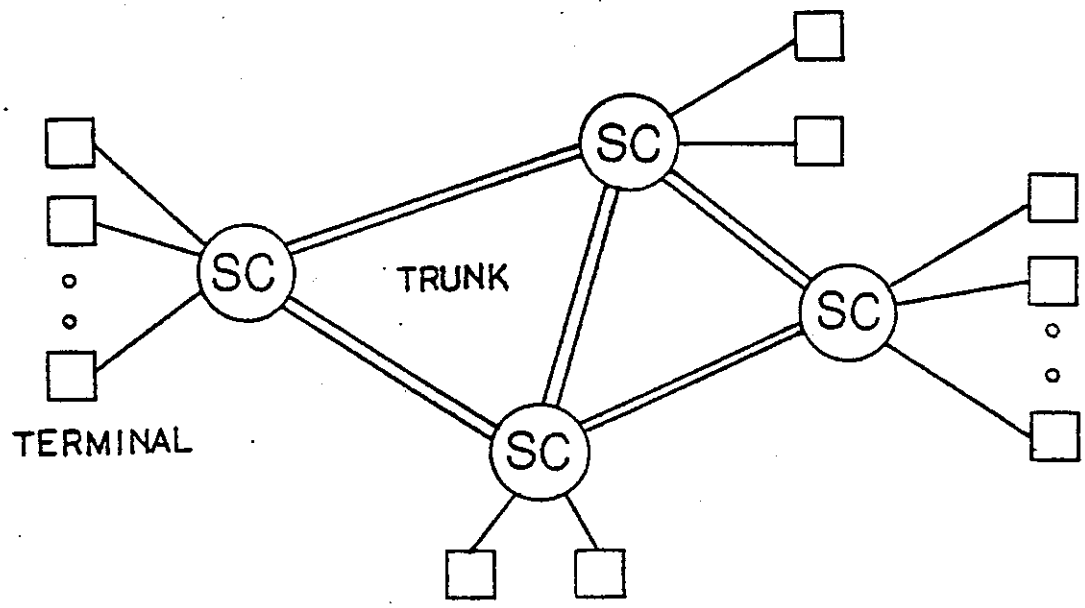


FIGURE 1.1.1 COMPUTER COMMUNICATION NETWORK

There are basically two kinds of switching techniques, line-switching and message-switching. The dial telephone system is a typical line-switched system, while

the telegraph system is a message-switched system. Brief descriptions of a line-switched network, a message-switched network, and the new integrated switched network, a mixture of line and message switching, follow.

1.1.1. Line-Switched Network

In a line-switched system, information from a sending terminal is not transferred to a receiving station until the network has set up a connection. As figure 1.1.2 shows, a terminal starting a call submits a "send request" to the exchange, where further dialing information is started. The nodal control processor generates an inquiry signaling message. This inquiry message precedes, link by link, to the receiver's exchange. There, a response message which reflects the status of the receiver desired is prepared. The response proceeds, again, link by link.

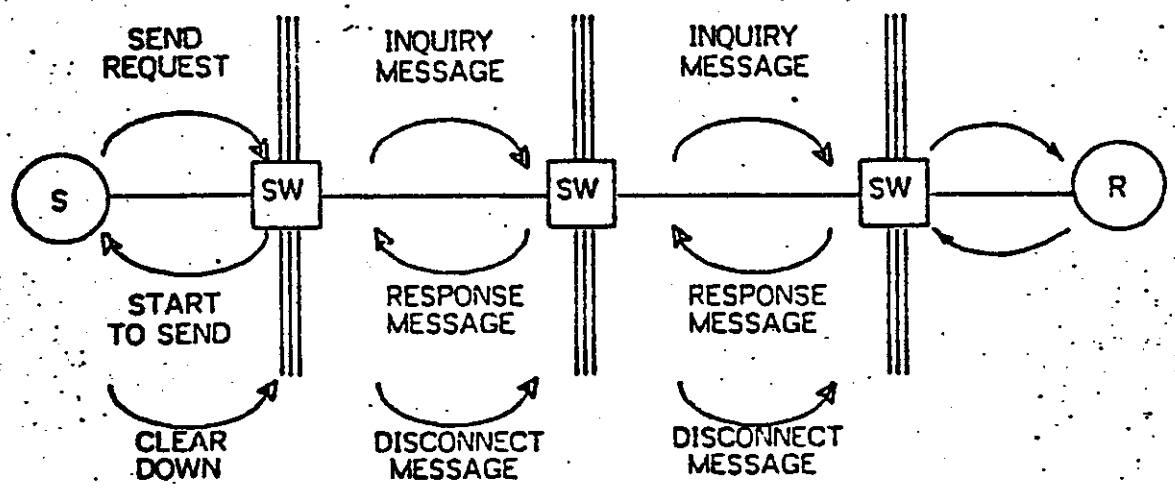


FIGURE 1.1.2 LINE-SWITCHED NETWORK

There are two possible ways to set up the path. The first is the forward path set-up, in which the inquiry message causes the nodal processors to connect the links

of the path from the sender to the receiver exchange, without the knowledge of the receiver status. The second is the backward path set-up which is initiated by the response message when the receiver is available.

In both instances, when the path is completed, a "start-to-send" message is sent to the sender by the sender's switching center. Release of a path is initiated by a "clear-down " command. During these two control signals, the path is reserved for this connection, and the switching processor, which is not affected by presence or absence of information flowing on the path, will not intervene.

In line-switched networks, signaling messages may be exchanged between nodes in two ways: via a special, common signaling channel exclusively dedicated to the transfer of network signaling messages, or over the same channel which also conveys the customer message (individual or separate signaling channel system). The latter automatically implies the forward path set-up from the sender to the receiver.

In principle, path establishment in a line-switched network is a store-and-forward process. Customer message transfers are always preceded and ended by a store-and-forward signaling phase.

1.1.2. Message-Switched Network

In a message-switched system, messages are temporarily stored in the nodes. Message transfer involves three steps: from the sender to his switching center, between switching centers, and from the destination switching center to the receiver. Figure 1.1.3 shows the typical signaling procedure. Customer messages are routed to the destination node with the help of address information contained in a header tagged to the message. It is assumed that the receipt of a message is signaled by an

acknowledgement message. Two kinds of network signaling messages can thus be distinguished: headers and pure signaling messages, such as acknowledgements.

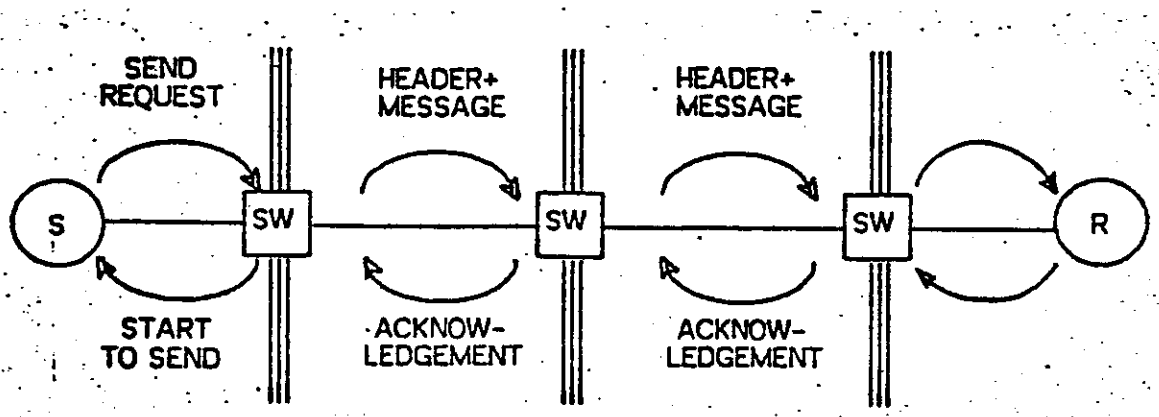


FIGURE 1.1.3 MESSAGE-SWITCHED NETWORK

Figure 1.1.4 shows the transfer delay of a line-switched network and a message-switched network. When the transmission time of customer messages over the path is long compared to the connect and disconnect time, line-switching is preferable. Line-switching has comparatively little overhead, because it does not need the large buffer space required for message-switching. On the other hand, when the size of the messages decreases, the overhead of line-switching increases, making message-switching preferable, since the total signaling of path connection for line-switching has more overhead than a node by node, store-and-forward message transmission. As long as the buffer space for message-switching is not too large, message-switching is more cost effective.

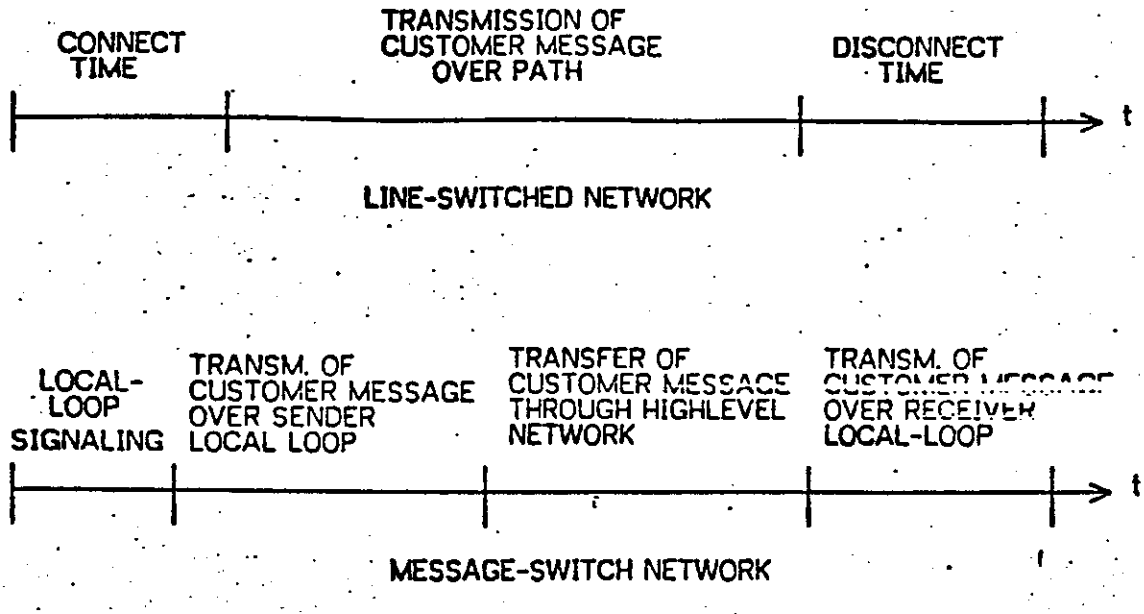


FIGURE 1.1.4 TIMING CONFIGURATION

There is a variation of the message switching technique called packet-switching. Packet-switching function and control properties are the same as those of message switching, but there is an upper limitation on the size of packets.

A message that is too large for a packet is broken into several packets, each with its own destination and sending information. The disadvantage of packet-switching is that it involves more overhead than ordinary message-switching. Instead of one overhead message header per a message, there are several packet header containing similar addressing information. The advantages of packet-switching are smaller buffer size requirement and more flexible flow control. ARPANET and AUTODIN are two of the many large resource sharing networks using packet-switching technique.

1.1.3. Integrated Switched Network

The trends of future requirements for digital communication systems indicate an increasing diversity of traffic characteristics such as:

(i). wide disparities between traffic rates, ranging from low-rate TTY terminals, requiring hundreds of bits/sec, to wideband video and graphics terminals, requiring hundreds of kilobits/sec.

(ii). wide disparities in transaction sizes, starting with interactive messages of several hundreds of bits and continuing up to bulk data transfers of millions of bits.

(iii). varying delivery times to accomodate voice and video, which requires continuous, near real-time response; interactive data terminals, which require response time in the order of seconds; and bulk data, which can be queued for hours.

From the description of the traffic requirements, it is clear that no single switching technique can give satisfactory response to all requests. A line-switched network will have tremendous overhead, or waste, on a communication facility designed for interactive data requisition. A simple calculation by James Martin [Mart 72] shows that only 0.1% of the total communication facility is used when a teletype is connected to a computer through a telephone line. Telephone systems use line switching, while the data generated by a teletype is a sequence of bursts, which is ideal for message- or packet- switching. On the other hand, neither message- nor packet- switching can handle voice communications or wideband video signals with the response time needed to maintain the integrity of the information. Because they posses sufficient redundancy, voice or video signals are not sensitive to the error rate of channels. Timing, however, is very critical, often in the order of tens of milliseconds of total path delay, and it is very difficult for a message-switched network, with its error detection and acknowledgement mechanisms, to meet such demands.

In order to use communication facilities more efficiently, it is nature to suggest the merging of different switching techniques into one network. Recently, several efforts [Forg 75, Covi 75] have been made to define digital communications schemes in which the network capacity is shared between a line-switched and a message-switched system. Such schemes allow the most efficient and appropriate switching technique to be chosen for each service request. Almost all these schemes need more processing power than that provided by the conventional network switching. However, as processor and memory costs continue to decline faster than transmission costs, an integrated switch becomes feasible and cost effective.

1.2. SENET Configuration

SENET (Slotted Envelope NETwork) is a proposed scheme for an integrated switching network. The idea is suggested by Coviello and Vena [Covi 75], and its implementation is discussed in [Barb 76] in more detail. SENET is a Time Division Multiplex (TDM) scheme in which time slices of fixed size, a frame period, are partitioned and allocated to the transmission of digitized voice and data packets. The voice component of a frame is further divided into slots allocated to ongoing line switching communications. Slots reserved along a path on the network establish a virtual line-switch path between the end subscribers.

This scheme can handle three different types of traffic.

Class I. Characterized by long transactions requiring continuous real time response (voice, video, facsimile). The transmission rate may vary from thousands of bit/sec (voice) to hundreds of thousands bits/sec (video).

Class II. Characterized by short discrete transactions requiring near real time response (interactive data, with data requisition being the typical example). The response should occur within seconds.

Class III. Characterized by long transactions requiring neither continuous nor immediate response (bulk data).

1.2.1. Frame

The detail of a frame is shown in figure 1.2.1. Starting at the 12 o'clock position, a certain number of bits are reserved for CCIS (Common Channel Interswitch Signaling). Following this region is a Class I region containing the real time traffic. The

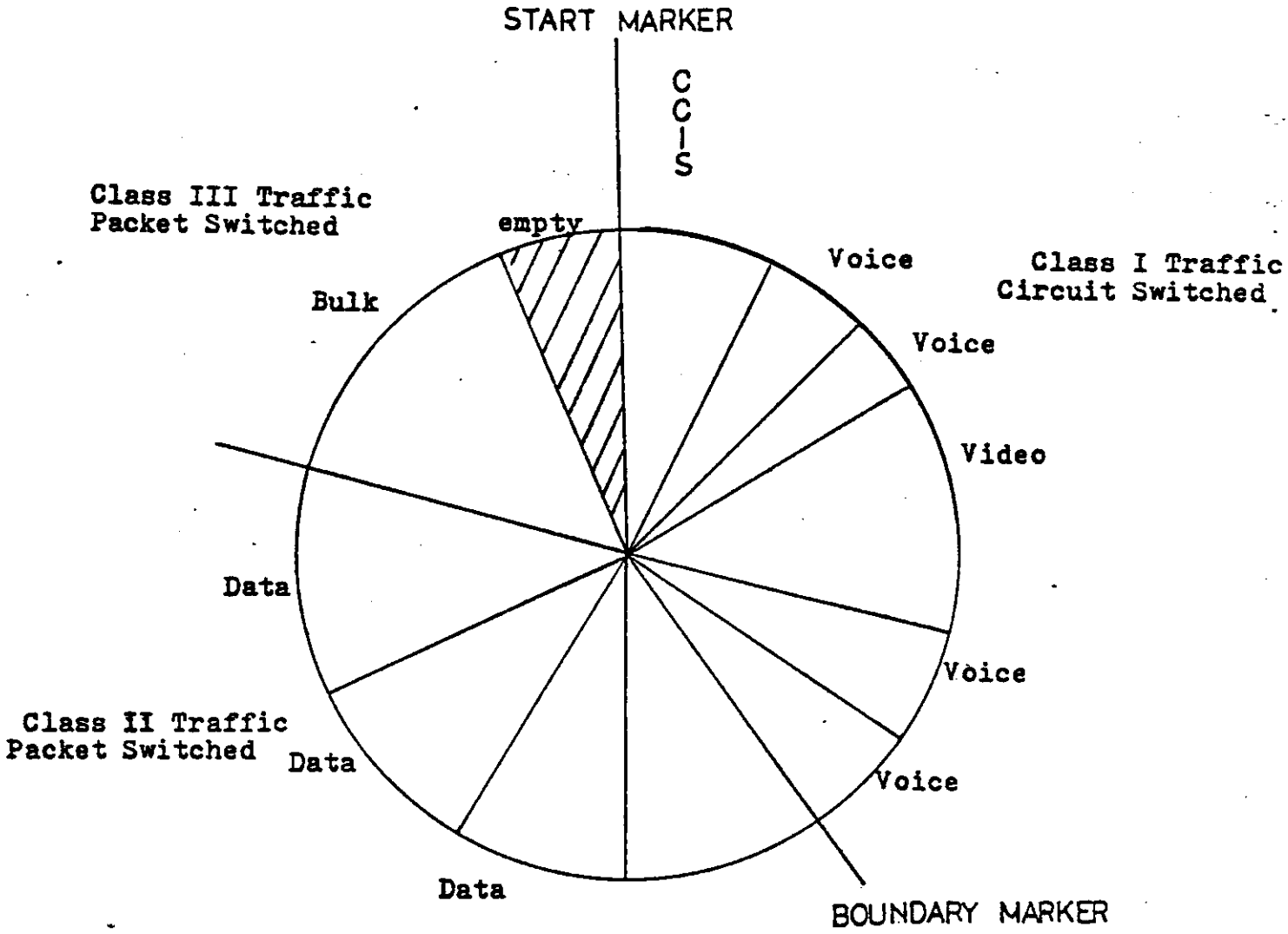


FIGURE 1.2.1 INTEGRATED SWITCHED NETWORK FRAME

end of the Class I region is indicated at 5 o'clock. Class II and Class III regions containing the interactive and bulk traffic respectively occupy the rest of the frame. All the interactive data or bulk data are transmitted inside the Class II region in disjoint packets. (For most of the thesis, we shall make no distinctions between Class II and Class III traffic. Unless we explicitly indicate otherwise, we will use the term Class II to indicate both interactive data and bulk data transmitted by packet switching.) Real time traffic, on the other hand, is transmitted inside the Class I region in fixed slots, with one slot allocated for each logical channel currently in use.

The scheme does not assume the existence of a master network clock. Each link transfers frames at a constant rate, but the links are not necessarily synchronized among themselves. Frame timing for each link is discussed in Chapter 4. In addition, the network does not require a homogeneous link of the same speed. For a given frame period, say 10 milliseconds, the number of bits contained in the frame depends on the speed of the link. Thus, for a T1 carrier, a 10-millisecond frame contains 15,440 bits, while for slower carrier, a 10-millisecond frame is smaller.

1.2.2. Service Requirements

Each of the three types of traffic requires different type of service. Class I is either accepted or rejected, with short connection delays and without error control. It represents a loss system in which there is a small probability of a connection being refused if no logical channel can be established between subscribers. Class I traffic requires low delay and a constant throughput in order to maintain the intelligibility of the message. Class II traffic is always accepted, but it may incur a system delay with short cross-network delay and reasonable response time. This traffic is characterized

by bursts of information followed by "waiting" periods, requiring high degree of reliability. Class III traffic is always accepted with longer cross-network delay than the previous class. Class II traffic also requires a high degree of reliability.

The real time requirements of Class I traffic dictate that it be processed in a special fashion by the integrated switch. Since voice or video signals carry a significant amount of redundancy, a moderate error rate can be tolerated in most applications. Elimination or reduction of the need for error control enables Class I traffic to be transmitted in a straightforward manner. The establishment of a logical path between two subscribers is reflected, on each switch along the path, by the updating of Class I Routing Tables internal to the integrated switches. These tables indicate, for each slot in the Class I region of an input frame, both the output frame (output channel) and the slot position reserved for the logical path.

Entries in the Routing Table are reserved according to the connecting request in the line switched system. The inquiry control messages proceeds, link by link, to the receiver. These messages also carry information about the data rate, slot size, and priority of the logical channel. The reservation of the slots along the communication path guarantees a fixed bandwidth for this type of traffic. The common characteristics of this type of traffic are : (i) long holding time, which is defined as the duration between the commands of connection and disconnection and is in the order of minutes, (ii) relatively less bits/frame compared to the size of one Class II packet which is always transmitted within one frame. Typically Class I slots require a few hundred of bits/frame, while Class II packets require in the order of thousand bits/frame.

The communication facility is divided into frames, but the transmission and the receiving of Class II traffic are packet oriented. The control functions, error checking,

buffer assignment, and acknowledgement generating are independently processed for each packet. The processor (or processors) assigns packet buffers to each incoming packet and copies the packets into the buffers, after an input frame has been completely written in the input frame buffer. In packet switching, at least one good copy of the data packet must exist in the communication system. For every correctly received packet, an acknowledgement is echoed back to the sender, so that the copy of the packet still at the sender can be released. The same philosophy applies to the integrated switch: A buffer will be released only if the packet is sent out to the next node and a positive acknowledgement of the packet is echoed back. The frame buffer is ready to accept a new input frame after copying the data packets into buffers and copying the voice slots directly into the corresponding output frame. In the meantime, various tasks such as error checking, routing, and controlling are processed on a per packet basis.

1.3. Previous Work

A considerable amount of research has been done during the last decade in the field of computer-communication networks. This research covers areas such as modeling, analysis, design, data evaluation, and measurement for the whole network and also for the communication processors.

A separate data-communication device for computers is considered after the development of large multi-access systems when the data traffic became crowded. Initially, the access terminals were few and close and the communication costs were not an important factor. The system needs a multiplexer and a demultiplexer to get efficient data access and good response time. The behavior of a local communication network is discussed by Chu [Chu69, Chu72].

The sharing on large data bases results in many terminals relatively far away and the communication costs gradually became dominant. There are two general models of a centralized computer-communication systems: the star network [Dol69] and the loop network [Pie72]. The data or requests to and from the terminals are no longer sent without any identification. The information is packed into packets or slots. The multi-drop line system [Cha72b] and ring switched data transmission system [Gra71, Hay71] are two typical examples. There is a header for each packet which contains sender and destination addresses and sometimes other control information.

For the more sophisticated switching functions, semi-independent or totally independent front-end processors are used rather than the hard wired multiplexers. Chang [Cha72a] presents a typical design and evaluation of such communication processors.

In order to achieve resource sharing in a larger scope [Rob70], instead of the communication between computer and terminals, the need of communication between computers arises. The problem of packet routing arises in such communication systems along with other control functions and procotols. A totally independent communication backbone system is implemented by communication processors. The ARPANET IMP [Hea70] is an example of an independent switching processor. Modeling the IMP as a queue server is quite successful for the analysis of the whole network. Both infinite queueing space and finite queueing space have been analyzed [Kle74a, Kle74b, Mei71]. As the cost of the processing power and memory decline more rapidly than that of the communication facility, the integration of the voice communication system becomes feasible and cost effective. The ideas of such exposition are discussed by Forgie and Coviello [For75, Cov75]. The implementation of an integrated switch has been tried on a multi-processor system [CMU75, Bar76] and in an associate processing system [Wal74].

The two different natures of voice and data make the analysis of an integrated switch very difficult. Kummerle [Kum744] used an approximation technique to describe the behavior of integrated switch. Later Bhat and Fischer [Bha75, Fis75] modeled the system as a pre-emptive multi-channel system. The number of simultaneous equations to be solved in the algorithm grows in the order of $s(s+1)/2$, where s is the number of channels. In Chapter 2, we model the system another way, and the number of simultaneous equations grows only in the order of s. A diffusion approximation technique is used in the same chapter. This technique has been used in many complicated queueing networks, [Fel66, Kob74a, Kob74b, Gav68]. The power of diffusion approximation for the time-dependent system is especially described by Gaver [Gav68].

The buffer behavior was discussed by Chu[Chu69, Chu72] in the early stage of multiplexer type communication processor systems. Later studies on packet-switching have dealt with the problem of packet size, buffer assignment, and the procotol of packet flow [Whi70, Rud72, Chu73, Clo73, Ros75]. Queueing network analyses were also undertaken. Gordon and Newell [Gor67] discovered the product form steady state distribution of a closed queueing network with exponential servers. Later Baskett et al [Bas75] extended this product form to more complicated networks: open , closed, or mixed, with different classes of customers. The service time may also have a more generalized distribution under some service disciplines, e.g. service time of rational Laplace Transform for processor sharing servers. The algorithm to calculate the product form solution is derived by Buzen [Buz73]. For more general service distribution, Chandy et al. [Cha75a, Cha75b, Her75] developed an approximation algorithm using Norton's theorem in electrical circuit theory. The analysis of closed, finite queueing networks with time lags was introduced by Posner[Pos68].

The problem of the network design was first studied by Kleinrock [Kle72]. The optimal capacity assignment problem was analytically solved for the linear link cost case. The criterion used was the total average waiting time. Meister et al. [Mei71] solved the problem with a different criterion: minimization of the total average of kth order of the waiting time. Using dynamic programming techniques, Frank et al. [Fra71] solved the discrete capacity assignment for a centralized S/F (Store and Forward) network. An exact, but very cumbersome, LP solution to the deterministic routing problem is proposed by Frank et al. [Fra72]. The same problem was solved more efficiently by Cantor et al. [Can72], who used decomposition techniques. The difficult problem of simultaneous routing and capacities assignment was approached, with mathematical programming techniques, by Gerla [Ger73].

1.4. Synopsis of Thesis

This thesis emphasizes the special SENET implementation of the integrated switch. An important characteristic of the integrated switch is that the voice holding time is much longer than the data packet service time. Because of this property, the waiting length of data packets will vary greatly under different loading of voice calls. This phenomenon is discussed in Chapter 2 in detail. Another important characteristics of the SENET implementation of integrated switch is the frame structure. Some influences of the frame structure on the buffer management are discussed in Chapter 3. The skew assignments due to the SENET frame structure are studied in Chapter 4. Characteristics such as protocol, flow control, and dynamic routing are not discussed in this study.

In Chapter 2, the buffer space, or the memory of the switching node, is assumed to be very large or infinite. The congestion of data packets, related to the slow change of voice traffic, is studied. First, some of the queueing models dealing with two kinds of job streams are discussed and compared with the integrated switch. Then the integrated switch is modeled and solved as a M/M/Y queue. Some difficulties in the algorithm are discussed, and an approximation algorithm is suggested. The integrated switch can also be interpreted as sequences of time-dependent processes. The diffusion process is used to approximate the switch. Finally, the results of different algorithms are compared.

The performance study of the switching processor, described in Chapter 2, suggests that the infinite memory space may not be a good assumption. In Chapter 3, some buffer management schemes are suggested and tested for the finite memory

space assumption. Three memory management schemes are discussed: division of memory into maximum size buffers, division of memory into different fixed size buffers, and dynamic allocation of memory space into buffers. Then a model of packets flowing in the network is studied, and the behavior of the buffers allocated to packets is investigated. Because of the congestion behavior of data packets, secondary storage is modeled into the integrated switch, and the effect of the secondary storage is studied in Section 3.4. In the last section of Chapter 3, a comparison of the above buffer managements for the integrated switch is discussed.

Because of the frame structure of the SENET implementation of the integrated switch, two special network design problems arise: capacity and skew assignment. Neither of the problems can be solved analytically, so some mathematical programming techniques are discussed. The capacity assignment problem is solved by a Coordinate Descent algorithm and multi-dimensional Newton's method. The skew assignment problem is formulated as a MILP (Mixed Integer Linear Program). Using graph theory, a heuristic program is developed. The heuristic program can find a local minimum but not a global one. The details of formulation and algorithm are discussed in Chapter 4, and results of each algorithm are compared.

The interpretation of the models and results is discussed in Chapter 5. Some suggestions for further studies and discussion of open problems are also included in this chapter.

CHAPTER 2 Performance Analysis of an Integrated Switch Processor

2.1. Introduction

In this chapter, the performance of the switching processor of an integrated data/voice communication network will be analyzed. Voice is transmitted by a virtual line-switch mechanism, and data is transmitted by packet-switching. There are many difference between data traffic and voice traffic. Voice traffic does not require error checking, acknowledgement, and dynamic routing. However, it has very strict real-time requirements. For data traffic, the processing requirements are almost exactly the opposite. The difference emphasized here is the service time requirement. For voice, there is a relatively long holding time. The average holding time of a public telephone system, for example, is in the order of three minutes. For data packets, the service time depends on the service ability of the system and is, in general, in the order of milliseconds. Although data traffic sometimes includes bulk messages which have millions of bits and require minutes of service time, those message are broken into packets, and each packet is processed independently. Transmitting packets of a large message is similar to transmitting packets for many small messages. Thus, our analysis of a system with two kind of streams must include consideration of the service time requirements of each kind of stream.

Besides having different service time requirements, voice (Class I) and data (Class II) traffic also have different service disciplines and different criteria for grades of service. (We will use Class I and Class II traffic instead of voice and data, for facsimile traffic has characteristics similar to voice and is treated as voice traffic.) If a

Class I job request arrives and cannot be serviced immediately, it is blocked and disappears without being serviced. For Class I jobs, the grade of service is determined by the blocking probability. Class II jobs can always be queued and wait for service if the service is not immediately available. Hence the grade of service for Class II jobs is determined by the mean waiting time for jobs.

The system becomes very complicated when the above service disciplines are considered. Kummerle [Kum74] was the first to give an approximation algorithm for estimating the average number of Class II packets in the system. Fisher and Harris [Fis75] did an analysis which assumed that the number of Class I jobs in the system is independent of each other from frame to frame. Later Bhat and Fisher [Bha75] used another approach which assumed that the two classes of jobs compete with each other for s channels. In their analysis, no channel is reserved only for Class I jobs.

In Section 2 of this chapter, we will present several queueing models and try to interpret the performance of those simple models as well as their similarity to the integrated switch. It is our goal here to establish some intuitive understanding of the integrated switch performance. In Section 3, an algorithm is given to solve the performance model of the integrated switch. In Section 4, the computation difficulty is discussed, and a simple example of error analysis is given. A simple conditional mean approximation algorithm is also suggested in this Section. In Section 5, the diffusion approximation for a finite time, overloaded system is developed. In Section 6, all the results of different algorithms are compared.

2.2. Models

Several queueing models applicable to analyzing a communication processor are discussed in this section. First some notation is introduced. Let:

λ = input rate of jobs. [jobs/second]

μ = service rate. $1/\mu$ is the mean service request per job.

c = channel capacity or the service ability of the switch.

T = average waiting time.

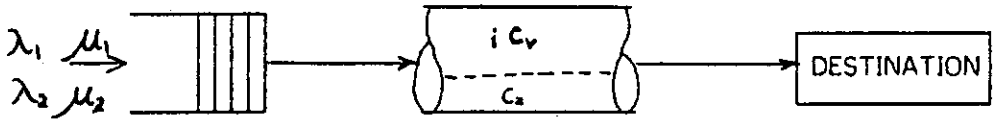
ρ = traffic intensity = $\lambda/c\mu$.

A subscript implies the class of job, e.g., λ_1 is the input rate of Class I jobs, and c_2 is the channel capacity for Class II jobs.

Kendall's notation A/B/m/n is used for the following queueing models. For the first two parameters, "M" refers to the Markovian character of Poissonian arrivals and exponential service; "GI" refers to generalized independent interarrival time or service time. The third parameter, m, stands for the number of servers, and the fourth parameter, n, stands for the number of storage spaces in the queue. For the third and fourth parameters, any "X,Y,Z" implies that the number of servers or queueing spaces is a random variable. For example an M/M/1/1 queue is a queue with input of Poisson process, service time of Exponential distribution, one server and one waiting space.

Here the input rates of jobs for both classes are assumed to be a Poisson process, and the service time is exponentially distributed.

2.2.1. Integrated Switch Model



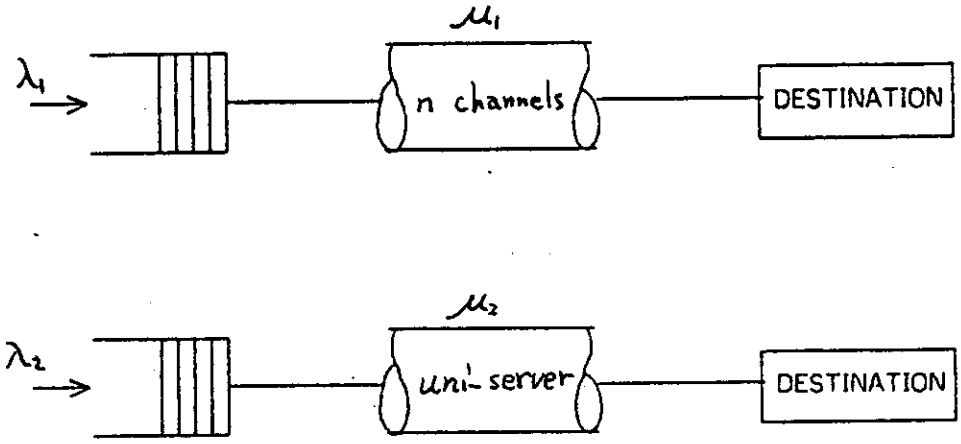
FIRGURE 2.2.1 M/M/Y QUEUEING MODEL

The whole service capacity of the switch is c, and each Class I job in the system will take some c_v service capacity out of the switch. If there are i Class I jobs in the system, the capacity to service Class II job is $(c-i*c_v)=c_2$, where c_v is the channel capacity needed to serve one Class I job. Let n be the maximum number of Class I jobs the switch will handle. Class I jobs cannot be queued, so n must be equal to or less than c/c_v. We will call this model the M/M/Y model for Class II jobs. Y(t) stands for a random process which represents the channel capacity of the system for Class II jobs at time t. We do not assume Y(t) is an independent sequence from frame to frame as was assumed by Fisher[Fis75]. Actually the auto-correlation coefficient between $Y(\tau)$ and $Y(t+\tau)$ is $1-0(\lambda_1\tau)$, where 0(x) is defined such that the limit of 0(x)/x remains bounded as x approaches zero. When $\lambda_1\tau$ is small, i.e., the average number of new voice calls per frame is small, Y(t) and $Y(t+\tau)$ are highly correlated, and the independence assumption will not be valid.

This model will be solved in the next section. Following are several queueing models dealing with two kinds of job streams. Each of the models has been solved by different researchers. We will give the model and the results of each analysis. These models will give a broad view of performance of queueing models with two kinds of job streams.

2.2.2. Models

(i)Segregated Line-Switched System



FIRGURE 2.2.2 SEGREGATED LINE-SWITCHED SYSTEM

There are two separate systems. One system consists of up to n lines allocating via a line-switching device, the other line involves message-switching and is modeled as an M/M/1 queue. The grade of service (GOS) for Class I jobs is the blocking probability of an M/M/n/n queue.

Let P_i be the probability of i jobs in the line-switched system. A simple Markov chain model can be built:

$$i*\mu_1 P_i = \lambda_1 P_{i-1} \qquad 2.2.1$$

GOS(1) = probability of a blocked job

= probability that there are n jobs in the line-switch system.

$$= P_n$$

$$= \frac{(\lambda/\mu)^n/n!}{\sum_{i=0}^{n}(\lambda/\mu)^i/i!} \qquad 2.2.2$$

The average waiting time for Class II jobs is :

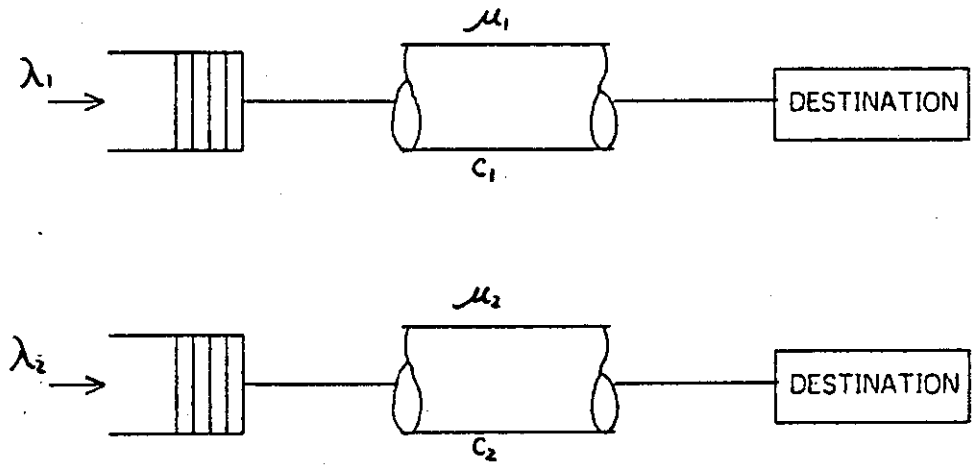
GOS(2) = average waiting time of Class II jobs

$$= 1/(c_2\mu_2-\lambda_2) \qquad 2.2.3$$

The above formula holds for $c_2\mu_2 > \lambda_2$. This condition is also necessary and sufficient for the above system to be stable and to have a finite average waiting time. Compared with the integrated switch, this system has exactly the same grade of service for Class I jobs and a lower grade of service for Class II jobs, as will be made clear later.

For all the following models, both Class I and Class II jobs may be queued for service, so the criterion for grade of service for both job classes is the average waiting time.

(ii)Segregated Message-Switched System



FIRGURE 2.2.3 SEGREGATED MESSAGE-SWITCHED SYSTEM

In this case, both Class I and Class II service are simple $M/M/c_i$ queueing system. Therefore, T_1, T_2, the average waiting time for Class I jobs and Class II jobs are:

$$T_1 = \frac{1}{c_1\mu_1 - \lambda_1} \qquad 2.2.4$$

$$T_2 = \frac{1}{c_2\mu_2 - \lambda_2} \qquad 2.2.5$$

where $\lambda_1 < c_1\mu_1$ and $\lambda_2 < c_2\mu_2$. $c_1 + c_2 = c$ is the total channel capacity. Define the average total waiting time T as:

$$T = \frac{\lambda_1}{\lambda} T_1 + \frac{\lambda_2}{\lambda} T_2 \qquad 2.2.6$$

where $\lambda = \lambda_1 + \lambda_2$ is the total input rate of jobs. The channel capacities c_1 and c_2 are assigned such that T is minimal. Using Lagrangian multiplier optimization technique, the optimum channel capacity assignment is as follows [Ver74]:

$$c_1 = \frac{\lambda_1}{\mu_1} + \sqrt{\frac{\lambda_1}{\mu_1}}/\beta \qquad 2.2.7$$

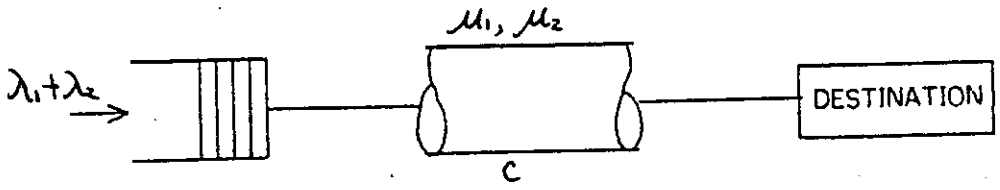
$$c_2 = \frac{\lambda_2}{\mu_2} + \sqrt{\frac{\lambda_2}{\mu_2}}/\beta \qquad 2.2.8$$

$$\text{where } \beta = \left(\sqrt{\frac{\lambda_1}{\mu_1}} + \sqrt{\frac{\lambda_2}{\mu_2}}\right)/\left(c - \frac{\lambda_1}{\mu_1} - \frac{\lambda_2}{\mu_2}\right) \qquad 2.2.9$$

This is the so called square-root channel capacity assignment[Ver74, Kle72], and

$$T = \frac{\lambda_1}{\lambda} \frac{1}{\mu_1 c_1 - \lambda_1} + \frac{\lambda_2}{\lambda} \frac{1}{\mu_2 c_2 - \lambda_2} \qquad 2.2.10$$

(iii)FIFO System



FIRGURE 2.2.4 FIFO SYSTEM

This system has only one channel with capacity c. The jobs come in and are serviced according to the FIFO (First In First Out) service discipline, regardless of their classes. The average waiting time is calculated by the formula of the M/G/1 queueing system [Ver74]:

$$T = \frac{1}{\mu c} + \frac{\sigma^2 \lambda \mu c + \lambda/\mu c}{2(\mu c - \lambda)} \qquad 2.2.11$$

where $\lambda = \lambda_1 + \lambda_2$

= total input rate of the system.

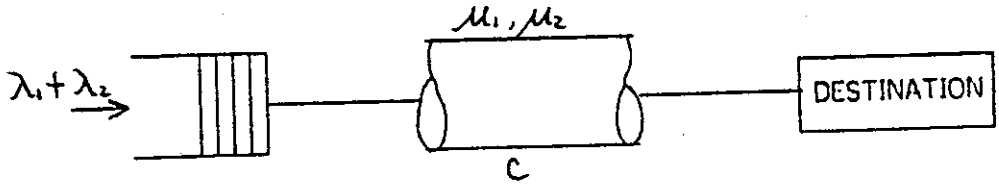
$$\mu = \lambda/(\lambda_1/\mu_1 + \lambda_2/\mu_2) \qquad 2.2.12$$

= the average service rate of jobs, the inverse of the mean service time.

$$\sigma^2 = [2\lambda_1/(\lambda\mu_1^2) + 2\lambda_2/(\lambda\mu_2^2) - (\lambda_1/\lambda\mu_1 + \lambda_2/\lambda\mu_2)^2]/c^2 \qquad 2.2.13$$

= the variance of the service time for the mixed stream.

(iv)Preemptive Priority System



FIRGURE 2.2.5 PREEMPTIVE PRIORITY SYSTEM

The Class I jobs have preemptive priority over Class II jobs. This system is

analyzed and solved by Avi-Itzhak and Naor [Avi61]. Because of the preemptive property, the service of Class I jobs is independent of the existence of Class II jobs. So from the formula of M/M/1 queue we will get:

$$T_1 = \frac{1}{\mu_1 c - \lambda_1} \qquad 2.2.14$$

The Class II jobs may be interrupted between service. The average waiting time is as follows:

$$T_2 = \frac{1/\mu_2 c + \lambda_1/[\mu_1 c(\mu_1 c - \lambda_1)]}{1 - \lambda_1/(\mu_1 c) - \lambda_2/(\mu_2 c)} \qquad 2.2.15$$

(v)Non-preemptive Priority System

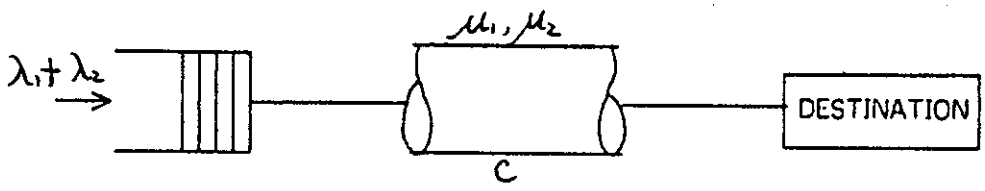


FIGURE 2.2.6 NON-PREEMPTIVE PRIORITY SYSTEM

The Class I jobs have priority over Class II jobs in the waiting line. This system is analyzed and solved by Cobham[Cob56]:

$$T_1 = \frac{1}{\mu_1 c_1} + \frac{\lambda_1/(\mu_1 c)^2 + \lambda_2/(\mu_2 c)^2}{1 - \lambda_1/(\mu_1 c)} \qquad 2.2.16$$

$$T_2 = \frac{1}{\mu_2 c} + \frac{\lambda_1/(\mu_1 c)^2 + \lambda_2/(\mu_2 c)^2}{\{1-\lambda_1/(\mu_1 c)\}\{1-\lambda_1/(\mu_1 c)-\lambda_2/(\mu_2 c)\}} \qquad 2.2.17$$

2.2.3. Comparison

The numerical results are plotted in Figures 2.2.7 and 2.2.8 There are two major points we would like to mention.

(I)The effect of long jobs. The average waiting time, T, is plotted in Figure 2.2.7, versus the service request of Class II jobs. The average service request of Class I jobs is assumed to be one unit in that figure. Class I jobs have priority over Class II jobs for priority systems. The average waiting time, T, increases very rapidly for the FIFO system and non-preemptive priority system with respect to the ratio μ_1/μ_2. The reason is that in these systems if a long Class II job is serviced, a long queue will be built up. There is a similar situation in the integrated switch model. If the service capacity for Class II jobs, $c_2=c-ic_v$, is less than λ_2/μ_2 and this situation lasts for a long period, a long queue will be built up. Thus, in the integrated switch model if $\lambda_2>(c-ic_v)*\mu_2$ for some i, the average waiting time will increase rapidly with respect to the ratio μ_2/μ_1, even if $\lambda_1/c\mu_1$, $\lambda_2/c\mu_2$ are kept constant.

(II)The effect of preemptive and non-preemptive priority. Figure 2.2.7 shows the total waiting time of jobs versus the length of low priority job service requests. Figure 2.2.8 shows the total waiting time of jobs with respect to the size of high priority job service requests. In Figure 2.2.8 the average service request of Class II jobs is assumed to be one unit, while the Class I jobs still have the priority. The figure shows that when the longer jobs have priority, there is little difference between non-preemptive priority and preemptive priority. The interrupted jobs are the smaller ones, which have little influence on the system behavior. The integrated switch model implies the preemptive priority of Class I jobs over Class II jobs. This is not true for the real SENET implementation which uses the non-preemptive priority scheme. However, the service requests of Class I jobs are several orders of magnitude higher than those of Class II jobs in the SENET implementation. From Figure 2.2.8, it is clear that we will expect little difference in performance between the real

SENET switch and its model. Of course in the integrated switch model, the Class I jobs will not be queued, and they will not block all of the channel capacity c. However, when the integrated switch is in overloaded state i, where the service capacity for Class II, $(c-ic_v)$, is less than the Class II job input rate, λ_2, a temporary queue with input rate less than its service rate will last a long period. This is the reason for the long queue in Figures 2.2.7 and 2.2.8 and will be the reason for the long queue of the integrated switch.

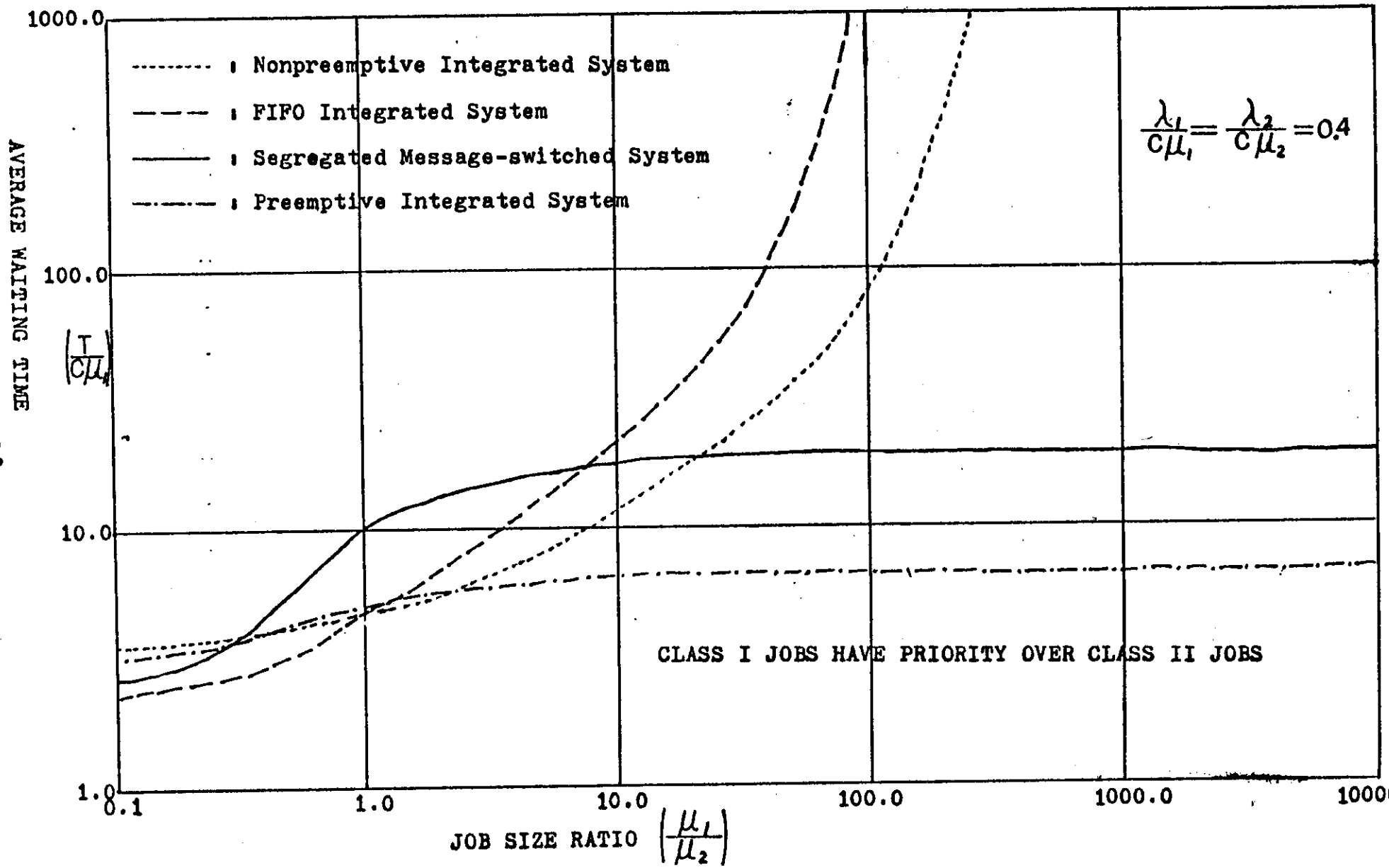


FIGURE 2.2.7 WAITING TIME WITH RESPECT TO JOB SIZE RATIO

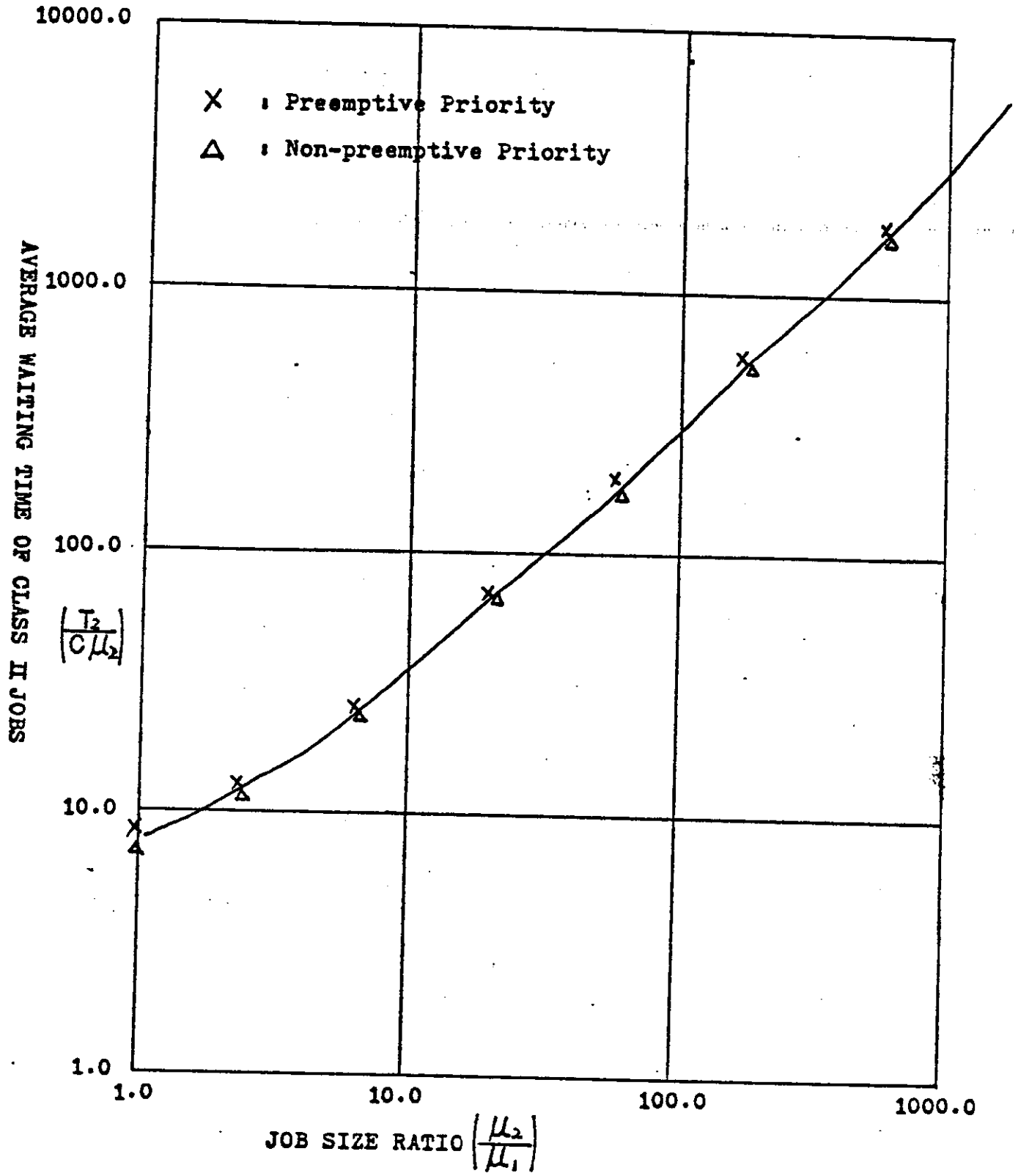


FIGURE 2.2.8 WAITING TIME WHEN LONG JOB HAS PRIORITY

2.3. Integrated Switch Model

In this section, an algorithm is suggested to solve the integrated switch model. By the memoryless property of the Poisson input processes and the independent exponentially distributed service requests, the model can be described as a Markov chain with state (i,j), where i is the number of Class I jobs and j is the number of Class II jobs in the system. The transition rate of the probability flowing out of the state (i,j) is $[\lambda_1+i\mu_1+\lambda_2+d_j\mu_2]$ and the transition rate of the probabilities flowing into the state (i,j) are λ_1, $(i+1)\mu_1$, λ_2, $d_j\mu_2$ from states (i-1,j), (i+1,j), (i,j-1), (i,j+1), respectively, for $0<i<n$ and $j>0$. Where $d_j=c-i*c_v$ is the channel capacity for Class II jobs when there are i Class I jobs in the switch. The non-boundary probability transition flow diagram is as follows:

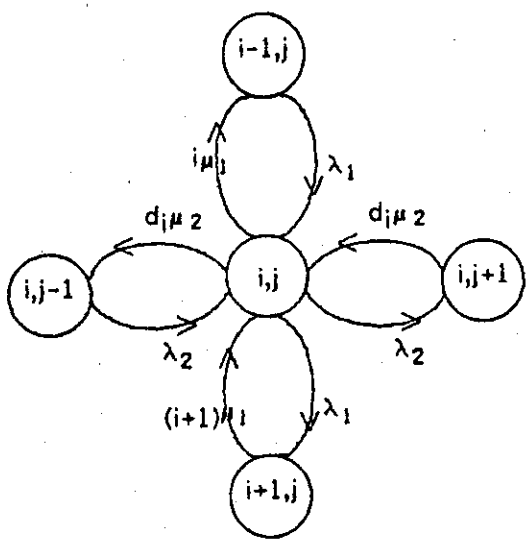


FIGURE 2.3.1 TRANSITION FLOW DIAGRAM

In the steady state, the probability transition rate flowing out of a state should be equal to the probability transition rate flowing into the state. The steady state distribution, $P_{i,j}$, will satisfy the following balance equations:

$$[\lambda_1 + i\mu_1 + \lambda_2 + d_i\mu_2] P_{i,j}$$

$$= \lambda_1 P_{i-1,j} + (i+1)\mu_1 P_{i+1,j} + \lambda_2 P_{i,j-1} + d_i\mu_2 P_{i,j+1} \qquad 2.3.1$$

for $0<i<n$, $j>0$.

For boundary states, some of the terms of equation (2.3.1) will vanish.

$$[\lambda_1 + i\mu_1 + \lambda_2] P_{i,j} = \lambda_1 P_{i-1,j} + (i+1)\mu_1 P_{i+1,j} + d_i\mu_2 P_{i,j+1} \qquad 2.3.2$$

for $0<i<n$, and $j=0$.

$$[\lambda_1 + \lambda_2 + d_i\mu_2] P_{i,j} = (i+1)\mu_1 P_{i+1,j} + \lambda_2 P_{i,j-1} + d_i\mu_2 P_{i,j+1} \qquad 2.3.3$$

for $i=0$, $j>0$.

$$[i\mu_1 + \lambda_2 + d_i\mu_2] P_{i,j} = \lambda_1 P_{i-1,j} + \lambda_2 P_{i,j-1} + d_i\mu_2 P_{i,j+1} \qquad 2.3.4$$

for $i=n$ and $j>0$.

$$[\lambda_1 + \lambda_2] P_{i,j} = (i+1)\mu_1 P_{i+1,j} + d_i\mu_2 P_{i,j+1} \qquad 2.3.5$$

for $i=0$ and $j=0$.

$$[i\mu_1 + \lambda_2] P_{i,j} = \lambda_1 P_{i-1,j} + d_i\mu_2 P_{i,j+1} \qquad 2.3.6$$

for $i=n$ and $j=0$.

2.3.1. Class I Jobs

Summing over equations (2.3.1) and (2.3.2) yields

$$[\lambda_1 + i\mu_1 + \lambda_2 + d_i\mu_2] P_i = \lambda_1 P_{i-1} + (i+1)\mu_1 P_{i+1} + \lambda_2 P_i + d_i\mu_2 P_i$$

or

$$[\lambda_1 + i\mu_1] P_i = \lambda_1 P_{i-1} + (i+1)\mu_1 P_{i+1} \qquad \text{for } 0<i<n \qquad 2.3.7$$

where $P_i = \sum_{j=0}^{\infty} P_{i,j}$.

$$\lambda_1 P_0 = \mu_1 P_1 \qquad i=0$$

and

$$n\mu_1 P_n = \lambda_1 P_{n-1} \qquad i=n$$

Substituting the above boundary condilions into equation (2.3.7) yields

$$\lambda_1 P_i = (i+1)\mu_1 P_{i+1} \qquad \text{for } 0 \le i < n \qquad 2.3.8$$

Together with the condition that P_i's are probability functions, i.e., $\sum_{i=0}^{n} P_i = 1$, P_i can be solved as:

$$P_i = P_0 \cdot \lambda_1^{\,i} / (i \,!\, \mu_1^{\,i}).$$

and

$$P_0 = [\, \sum_{i=0}^{n} \lambda_1^{\,i} / (i \,!\, \mu_1^{\,i}) \,]^{-1}.$$

G_1, the grade of service for Class I, will be:

G_1 = blocking probability for Class I jobs.

= a new Class I job comes in and finds the system at states (n,j).

= probability that system has n Class I jobs.

$$= P_n$$

$$= \lambda_1^{\,n}/(n!\mu_1^{\,n}) \,/\, [\sum_{j=0}^{n} \lambda_1^{\,j}/(j \,!\, \mu_1^{\,j})] \qquad 2.3.9$$

This equation is known as Erlang B loss formula [Sys60]. The grade of service for Class I jobs is the same as the grade of service for an M/M/n/n queueing system for line-switching. The Class I jobs have inherent preemptive priority over Class II jobs, so the system state of the integrated switch for Class I jobs is the same as in a system without any Class II jobs. Figures 2.3.2 and 2.3.3 show the numerical behavior of equation (2.3.9). The channel utilization factor u is defined as

$$u = \sum_{i=0}^{n} \frac{i}{n} P_i \qquad 2.3.10$$

2.3.2. Class II Jobs

For Class II jobs, the variation of the queue length is much more complicated. Define:

$$\pi_i(z) = \sum_{j=0}^{\infty} z^j P_{i,j} \,/\, \sum_{j=0}^{\infty} P_{i,j} \qquad 2.3.11$$

= generating function of Prob[Number of Class II jobs | i Class I jobs]

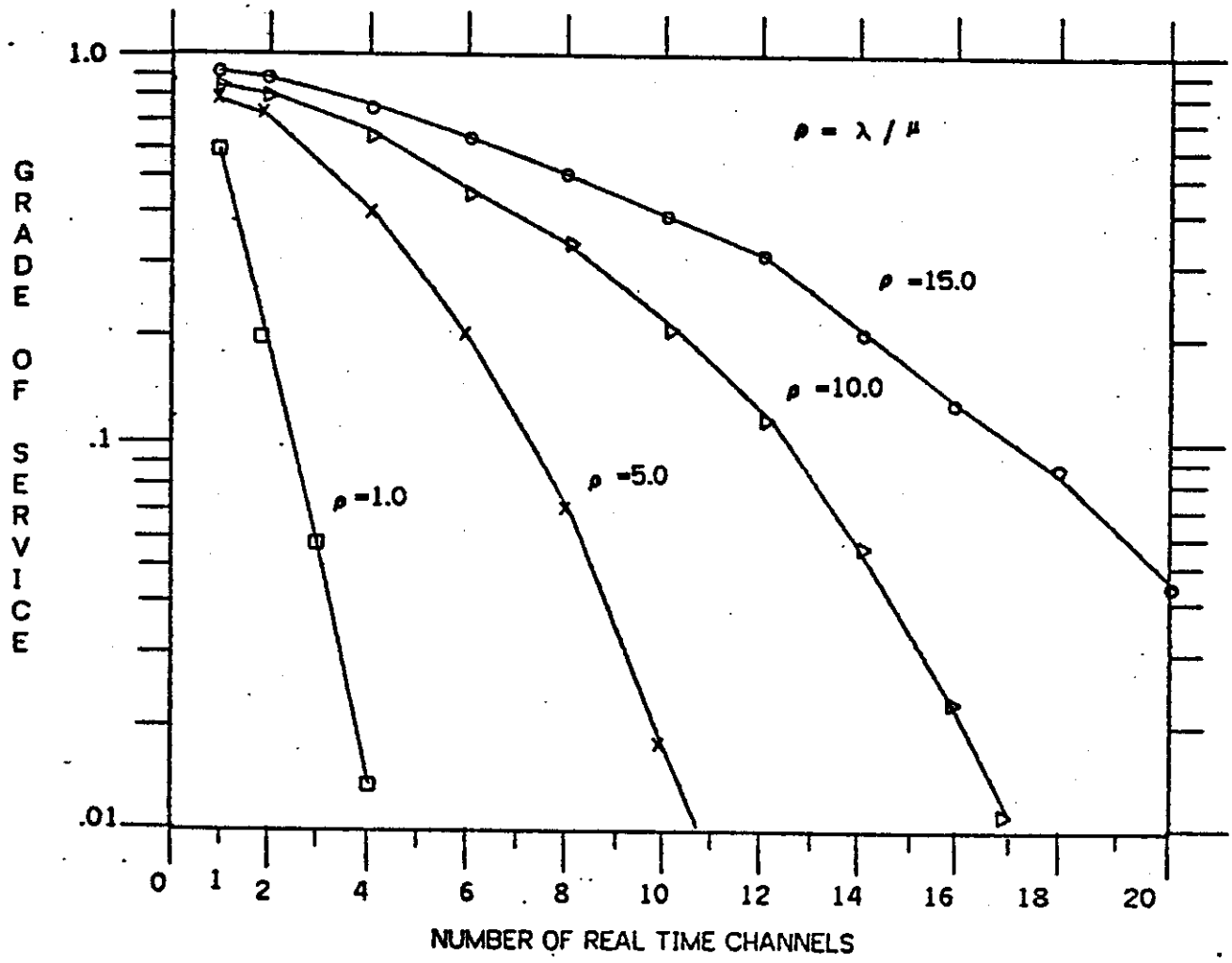


FIGURE 2.3.2 GOS OF CLASS I JOBS

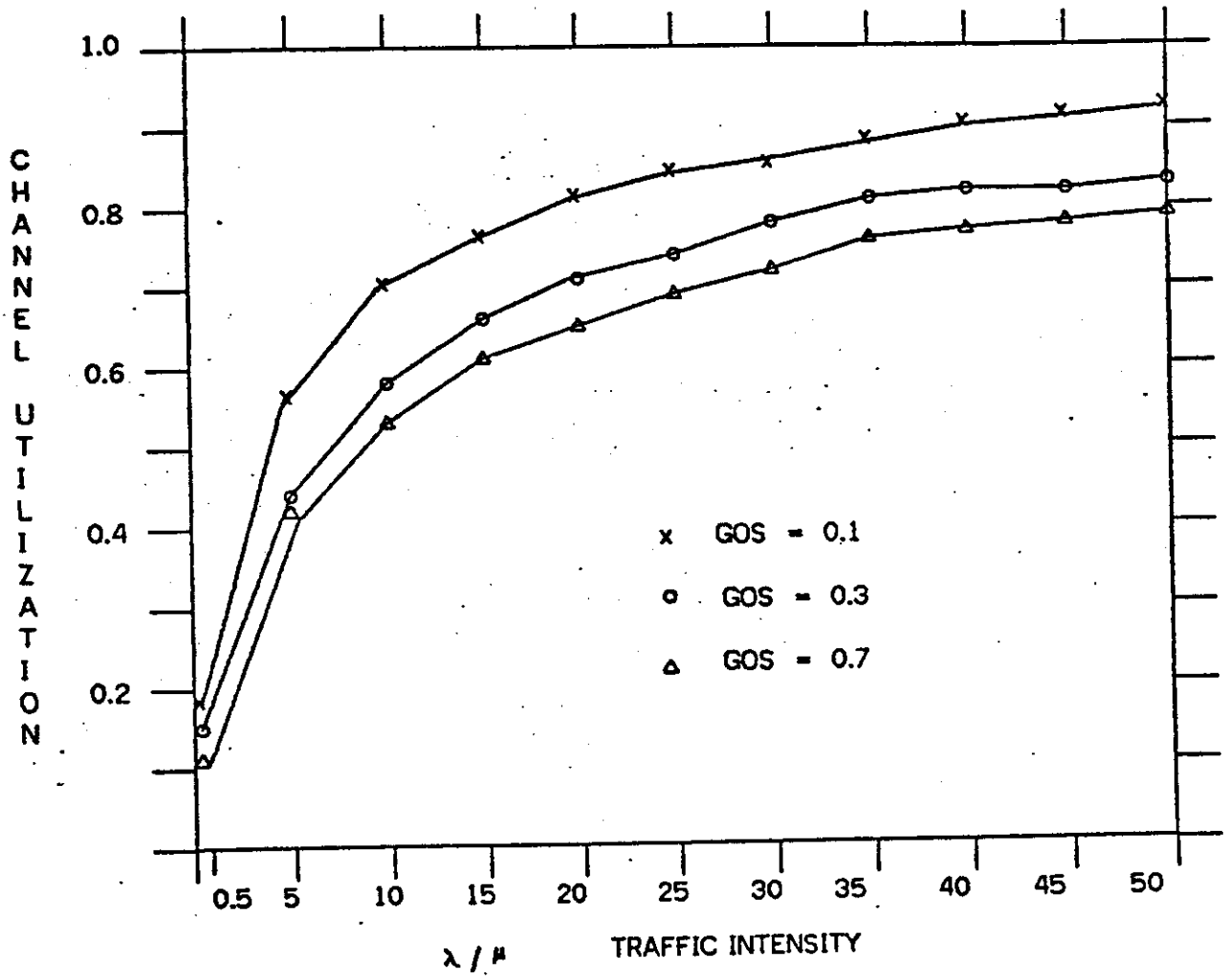


FIGURE 2.3.3 CHANNEL UTILIZATION OF CLASS I JOBS

= conditional generating function for Class II jobs with i Class I jobs.

Multiplying equation (2.3.1) by z^j and summing over all j, we have:

$$[\lambda_1 + i\mu_1 + \lambda_2 + d_i\mu_2]\,\pi_i(z)\cdot P_i$$

$$=\lambda_1\pi_{i-1}(z)P_{i-1}+(i+1)\mu_1\pi_{i+1}(z)P_{i+1}+\lambda_2 z\pi_i(z)P_i+d_i\mu_2/z\cdot\pi_i(z)P_i-d_i\mu_2(1-\frac{1}{z})P_{i,0}$$

Substituting equation (2.3.8) into the above equation yields :

$$[\lambda_1+i\mu_1+\lambda_2(1-z)+d_i\mu_2(1-1/z)]\pi_i(z) - i\mu_1\pi_{i-1}(z) - \lambda_1\pi_{i+1}(z)$$

$$= d_i\mu_2(1-\frac{1}{z})P_{i,0} / P_i. \qquad 2.3.12$$

for $0<i<n$.

Similar methods yield:

$$[\lambda_1+\lambda_2(1-z)+d_i\mu_2(1-1/z)]\pi_i(z) - \lambda_1\pi_{i+1}(z) = d_i\mu_2(1-\frac{1}{z})P_{i,0}/P_i. \qquad 2.3.13$$

for $i=0$,

$$[i\mu_1+\lambda_2(1-z)+d_i\mu_2(1-1/z)]\pi_i(z) - i\mu_1\pi_{i-1}(z) = d_i\mu_2(1-\frac{1}{z})P_{i,0}/P_i \qquad 2.3.14$$

for $i=n$.

To simplify the equations, define

$u_i(z)$ = coefficient of $\pi_i(z)$ of equation (2.3.12)

$$= \lambda_1 + i\mu_1 + \lambda_2(1-z) + d_i\mu_2(1-\frac{1}{z}) \qquad 0\le i<n \qquad 2.3.15$$

and

$$u_n(z) = n\mu_1 + \lambda_2(1-z) + d_n\mu_2(1-\frac{1}{z})$$

Similarly

b_i = coefficient of the right hand side of equation (2.3.12)

$$= d_i\mu_2\, P_{i,0} / P_i \qquad 2.3.16$$

Equation (2.3.13) and (2.3.14) can be rewritten in the matrix form as:

$$A(z)\,\pi(z) = (1-\frac{1}{z})\,b \qquad 2.3.17$$

where $\pi(z)$ and b are column vectors with ith element $\pi_i(z)$ and b_i, respectively.

$$\pi(z)=[\pi_0(z)\ \pi_1(z)\ .....\pi_n(z)]^T$$

$$b = [b_0\ b_1\ \ldots\ldots b_n\ ]^T$$

of dimension (n+1), where T stands for the transpose of a vector.

A(z) is a square matrix as follows:

$$A(z) = \begin{bmatrix} u_0(z) & -\lambda_1 & & & & \\ -\mu_1 & u_1(z) & -\lambda_1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & -i\mu_1 & u_i(z) & -\lambda_1 & \\ & & & \ddots & \ddots & -\lambda_1 \\ & & & & -n\mu_1 & u_n(z) \end{bmatrix}$$

We will solve the $\pi_i(z)$, or Prob[j | i], from equation (2.3.17). First b is solved from the following two properties of $\pi_i(z)$:

(i) $\pi_i(1) = 1$, for all i, because $\pi_i(z)$ is a generating function of a probability distribution. From equation (2.3.11)

$$\pi_i(1) = \sum_{j=0}^{\infty} P_{i,j} \Big/ \sum_{j=0}^{\infty} P_{i,j} = 1.$$

(ii) $\pi_i(z)$ are finite in the region $0 \le z \le 1$ for all i.

$$\pi_i(z) = \sum_{j=0}^{\infty} z^j P_{i,j} \Big/ \sum_{j=0}^{\infty} P_{i,j}$$

$$\le \sum_{j=0}^{\infty} P_{i,j} \Big/ \sum_{j=0}^{\infty} P_{i,j}$$

$$= 1 \qquad \text{for } 0 \le z \le 1$$

This implies that if $\pi_i(z)$ can be expressed as $\pi_i(z) = \frac{f(z)}{g(z)}$, then for all z_j such that z_j in (0,1) and $g(z_j)=0$ $f(z_j)=0$.

From the two conditions above, b can be determined. The algorithm is given below. First define a matrix R(z) with (i,j) element $r_{ij}(z)$ such that:

$$r_{ij}(z) = (-1)^{i+j} * a^*_{ji}(z).$$

where $a^*_{ji}(z)$ is the cofactor of $a_{ji}(z)$, the (j,i)th element of matrix A(z).

By the theory of matrices [Wyl60],

$$\sum_{j=0}^{n} r_{ij}(z) \cdot a_{jk}(z) = \sum_{j=0}^{n} (-1)^{i+j} a^*_{ji}(z)\, a_{jk}(z)$$

is the determinant of A(z) with ith column substituted by the kth column. If $i \neq k$, then this determinant has two identical columns, i and k, and the value of the determinant is zero. If i=k, the value is the determinant of matrix A(z). We have:

$$\sum_{j=0}^{n} r_{ij}(z) \cdot a_{jk}(z) = \delta_{ik} |A(z)| \qquad \text{for all } i,k \qquad 2.3.18$$

Following a similar argument, it can be proven that:

$$\sum_{j=0}^{n} a_{ij}(z) \cdot r_{jk}(z) = \delta_{ik} |A(z)| \qquad \text{for all } i,k \qquad 2.3.19$$

where δ_{ik} is the Kronecker delta, such that

$$\delta_{ik} = \begin{cases} 0 & i \neq k \\ 1 & i = k \end{cases}$$

Hence

$$A^{-1}(z) = \frac{R(z)}{|A(z)|}$$

where |A(z)| is the determinant of matrix A(z). So equation (2.3.17) becomes

$$\pi(z) = \frac{R(z)}{|A(z)|} \left(1 - \frac{1}{z}\right) b \qquad 2.3.20$$

In other words, $\pi(z)$, expressed in another form, is $\frac{f(z)}{g(z)}$ of a generating function. All zeros of the denominator in the region (0,1) must also be zeros of the numerator.

Theorem 1: |A(z)|=determinant of A(z) has exactly n different single roots in (0,1), and z=1 is also a root.

This theorem can be proven by using methods similar to those of [Mil68]. The basic argument follows: |A(1)|=0 is trivial, for the row summation of A is zero.

Define $g_i(z)$ as the principal minor of order i of A(z). Then $|A(z)|$ is equal to $g_n(z)$, a polynomial of degree (2n+2).

$$g_i(z) = \begin{vmatrix} u_0(z) & -\lambda_1 & 0 & 0 & 0 & 0 \\ -\mu_1 & u_1(z) & -\lambda_1 & 0 & 0 & 0 \\ 0 & \cdot & \cdot & \cdot\ \cdot\ \cdot & \cdot & 0 \\ 0 & 0 & \cdot & \cdot\ \cdot\ \cdot & u_{i-1}(z) & -\lambda_1 \\ 0 & 0 & 0 & \cdot\ \cdot\ \cdot & -i\mu_1 & u_i(z) \end{vmatrix}$$

Clearly,

$$g_i(z) = u_i(z)g_{i-1}(z) - i\lambda_1\mu_1 g_{i-2}(z) \qquad \text{for } i>1. \qquad 2.3.21$$

with $g_0(z)=u_0(z)$, and $g_{-1}(z)=1$.

By $u_i(0)\to-\infty$, $u_i(\infty)\to-\infty$, and $u_i(1)=\lambda_1+i\mu_1>0$. It is easy to see that the signs of $g_i(0)$ and $g_i(\infty)$ are $(-1)^{i+1}$, and $g_i(1)>0$ for all $i<n$.

From the above property, it is clear that there is one real root $z_1^{(0)}$ of $g_0(z)$ in (0,1), and one root in $(1,\infty)$, because $g_0(0)<0$, $g_0(\infty)<0$, and $g_0(1)>0$. Substituting $z_1^{(0)}$ into equation (2.3.21) yields

$$g_1(z_1^{(0)}) = u_1(z_1^{(0)})g_1(z_1^{(0)}) - \lambda_1\mu_1$$

$$= -\lambda_1\mu_1<0$$

and

$$g_1(0) <0,\ g_1(1) > 0.$$

So there is one real root of $g_1(z)$ in each of the regions $(0,z_1^{(0)})$ and $(z_1^{(0)},1)$, and there are two roots in the region $(1,\infty)$. By the same technique, counting the number of sign changes in the sequence $g_i(z)$, one can determine the number of zeros of $g_i(z)$ in (0,1). If $z_j^{(i)}$ is the jth root of $g_i(z)$, then the consecutive $g_{j+1}(z_j^{(i)})$ will have different signs. Thus $z_j^{(i+1)}$, roots of $g_{i+1}(z)$, will be in $(z_{j-1}^{(i)},z_j^{(i)})$ interval for j=0,1,..,i. The proof can be continued by the above interval partition procedure iteratively from i=1 to n. The number of roots of

$g_i(z)$ in (0,1) will be equal to (i+1) for all i<n. When i=n, one is also a root, the interval $(z_{n-1}^{(n)},1)$ does not have a root, and the total number of roots for |A(z)| in (0,1) is n. The same procedure is also carried out numerically to find all the roots. #

Theorem 2: $R(z_i)$ has rank one for all z_i of a root |A(z)| in (0,1].

Suppose that $\bar{r}_j$ is the jth row of matrix R(z) and that $\bar{a}_k$ is the kth column of A(z). Then from equations (2.3.18) and (2.3.19),

$$\bar{r}_j \cdot \bar{a}_k = \delta_{jk} |A(z)| \qquad \text{for all i,k.} \qquad 2.3.22$$

Now for $0<z\leq 1$ and |A(z)|=0, we will have

$$\bar{r}_i \cdot \bar{a}_k = 0 \qquad \text{for all i,k.} \qquad 2.3.23$$

|A(z)| only has a single root in (0,1), because there are n independent $\bar{a}_k$ out of (n+1) column of A(z). For any row vector $\bar{x}$ of dimension (n+1), if there exist n independent row vector $\bar{a}_k$ such that $\bar{x}\cdot\bar{a}_k=0$, then $\bar{x}$ has only one degree of freedom. For any $\bar{y}$ such that $\bar{y}\cdot\bar{a}_k=0$ for $1\leq k\leq n$, it is implied that $\bar{y}=\alpha\bar{x}$, for a scalar α. This is equivalent to saying that in a space of dimension (n+1) with n independent linear constraints, the solution will always be on a straight line, i.e., $y=\alpha x$. #

Because R(z) has a rank equal one for all $0\leq z_i\leq 1$ and $|A(z_i)|=0$, any row of R(z) can be chosen as $r(z_i)$ and satisfies the following equations:

$$r(z_i) \cdot b = 0 \qquad \text{for } 0\leq z_i<1,\ i=1,2,\ldots n \qquad 2.3.24$$

and

$$r(1) \cdot b = \frac{d}{dz} |A(z)|_{z=1} \qquad 2.3.25$$

Now that we have (n+1) unknown and (n+1) independent equations, b_i can be solved.

The element of $r(z_i)$ is the determinant of an n dimension submatrix of $A(z_i)$. Since $R(z_i)$ has rank of one, any row of R(z) can be used. If the zeroth row is chosen, the result is:

$$r_j(z) = \lambda_1^j \cdot \begin{vmatrix} u_{j+1}(z) & -\lambda_1 & 0 & 0 \\ -(j+2)\mu_1 & u_{j+2}(z) & -\lambda_1 & 0 \\ \cdots & \cdots & \cdots & 0 \\ 0 \;\; 0 & \cdots & u_{n-1}(z) & -\lambda_1 \\ 0 \;\; 0 & \cdots & -n*\mu_1 & u_n(z) \end{vmatrix}$$

Basically all $r_j(z)$ follow the three term iteration:

$$r_j(z) = \frac{u_{j+1}(z)}{\lambda_1} r_{j+1}(z) - \frac{(j+2)\mu_1}{\lambda_1} r_{j+2}(z) \qquad 2.3.26$$

Because there is one degree of freedom for all $z_i \neq 0$, we can set any $r_i(z)=1$ and calculate the rest of $r_j(z)$.

R(1) has a very special property: Not only is every row linearly dependent on each other, but every row is exactly the same. Thus for any row, the jth element of the row vector is as follows:

$$r_j(1) = (n!\,\mu_1^n) \cdot \lambda_1^j / (j!\,\mu_1^j) \qquad 2.3.27$$

because $u_j(1)=\lambda_1+j\mu_1$, and the summation of all row, except one, is zero.

In order to calculate equation (2.3.25), we must first calculate $\frac{d}{dz}|A(z)|$. The derivative of a determinant is the summation of determinants, each with a derivative on one row of the original matrix. So

$$\frac{d}{dz}|A(z)| = \sum_{i=0}^{n} |A_i'(z)| \qquad 2.3.28$$

where $A_i'(z)$ has elements that are the same as those of A(z), except for the ith row, which is the derivative of the ith row of A(z).

By a straightforward manipulation, it can be proved that:

$$|A_i'(z)|_{z=1} = (n!\,\mu_1^n) \cdot \lambda_1^i / (i!\,\mu_1^i) \cdot u_i'(1) \qquad 2.3.29$$

where $u_i'(z)$ is the derivative of $u_i(z)$ with respect to z.

$$\frac{d}{dz}|A(z)|_{z=1} = \sum_{i=0}^{n} (n!\,\mu_1{}^n) \cdot \lambda_1{}^i / (i!\,\mu_1{}^i) \cdot (-\lambda_2 + d_i\mu_2) \qquad 2.3.30$$

Substituting the $r_j(1)$ into the above equation, we have:

$$\sum_{j=0}^{n} r_j(1)\, b_j = \sum_{j=0}^{n} r_j(1) \cdot (-\lambda_2 + d_j\mu_2) \qquad 2.3.31$$

$r_j(1)$ is proportional to P_j, the probability of the system having j Class I jobs. Let $r_j(1)$ be normalized and be equal to P_j. There is a physical interpretation of the above equation. $b_j/(d_j\mu_2)=P_{j,0}/P_j=P_{0|j}$ is the probability that no Class II job is in the switch when there are j Class I jobs in the system. The left hand side of the equation, which is the summation of the probability that the channel has no Class II jobs multiplied by the service ability available, is equal to the whole wasted channel capacity. The second term of the right-hand side of equation (2.3.31) is the total service ability for Class II jobs. Thus the right hand side , which is the service abililty available minus the service requests, is also the total wasted channel capacity.

By the n homogeneous equations of (2.3.24) and the above non-homogeneous equation (2.3.31), b_j can be solved uniquely. All other system parameters can be derived from b_j. We next give an example of how to calculate $\pi_i'(1)$, the average queue length when there are i Class I jobs in the switch.

If we differentiate equation (2.3.17), $A(z)\pi(z)=(1-\frac{1}{z})b$, with respect to z, we get:

$$A'(z)\pi(z) + A(z)\pi'(z) = b/z^2 \qquad 2.3.32$$

or

$$A(1)\pi'(1) = b - A'(1)\cdot 1$$

where 1 is a vector with all the elements containing one. The ith row of the vector equation is:

$$\lambda_1[\pi_i'(1) - \pi_{i+1}'(1)] + i\mu_1[\pi_i'(1) - \pi_{i-1}'(1)] = b_i + \lambda_2 - d_i\mu_2 \qquad 2.3.33$$

for $0<i<n$.

$$\lambda_1[\pi_0'(1) - \pi_1'(1)] = b_0 + \lambda_2 - d_0\mu_2$$

and

$$n\mu_1[\pi_n'(1) - \pi_{n-1}'(1)] = b_n + \lambda_2 - d_n\mu_2$$

With b_i's known, there are (n+1) variables, $\pi'(1)$, but the (n+1) equations above are not linearly independent, for the determinant $|A(1)|$ is zero. An extra equation is needed to solve the $\pi'(1)$. Multiplying both side of equation (2.3.32) by $A^{-1}(z)$, $\frac{R(z)}{|A(z)|}$, we get

$$\pi'(z) = \frac{R(z)}{|A(z)|}[b/z^2 - A'(z)\pi(z)] \qquad 2.3.34$$

If we let z approach 1 and use the L'Hospital's Rule to calculate the resulting limit, we get:

$$\begin{aligned}\pi'(1) &= \{R'(z)[b/z^2 - A'(z)\pi(z)] \\ &\quad + R(z)[-2b/z^3 - A''(z)\pi(z) - A'(z)\pi'(z)]\} \,/\, [\tfrac{d}{dz}|A(z)|]_{z=1} \\ &= \{R'(1)[b - A'(1)\pi(1)] + R(1)[-2b - A''(1)\pi(1) - A'(1)\pi'(1)]\}/a_1 \qquad 2.3.35\end{aligned}$$

where $a_1 = \frac{d}{dz}|A(z)|_{z=1} = \sum_{j=0}^{n} r_j(1)(-\lambda_2+d_j\mu_2)$,

$A'(1)$ and $A''(1)$ are diagonal matrices with elements $(-\lambda_2+d_i\mu_2)$ and $(-2d_i\mu_2)$, respectively. Because all rows of R(1) are the same, $\pi_i'(1)$ can be written in the following form:

$$\begin{aligned}\pi_i'(1) &= \{\sum_{j=0}^{n} R_{ij}'(1)[b_j - \lambda_2 + d_j\mu_2] \\ &\quad + \sum_{j=0}^{n} r_j(1)[-2b_j - 2d_j\mu_2 - (-\lambda_2 + d_j\mu_2)\pi_j'(1)]\} \,/\, a_1 \qquad 2.3.36\end{aligned}$$

Because only one equation is needed from the above to solve $\pi'(1)$, let i equal zero. Then:

$$\pi_0'(1) = \phi_0 + [a_2 - \sum_{j=0}^{n} \gamma_j\pi_j'(1)]/a_1 \qquad 2.3.37$$

where

$$\phi_0 = \sum_{j=0}^{n} R_{0j}'(1)[b_j + \lambda_2 - d_j\mu_2] \,/\, a_1$$

$$\gamma_j = r_j(1)[-\lambda_2 + d_j\mu_2] = n!\,\mu_1^{\,n} \cdot \lambda_1^{\,j}(-\lambda_2 + d_j\mu_2) \,/\, (j\,!\,\mu_1^{\,j})$$

$$a_1 = \sum_{j=0}^{n} \gamma_j = \sum_{j=0}^{n} r_j(1)[-\lambda_2 + d_j\mu_2]$$

$$= n!\,\mu_1^n \sum_{j=0}^{n} \lambda_1^j \cdot (-\lambda_2 + d_j\mu_2) / (j!\,\mu_1^j)$$

$$a_2 = \sum_{j=0}^{n} r_j(1)[-2b_j + 2d_j\mu_2]$$

where $R_{0,j}'(1)$ of equation (2.3.36) is given by the derivative of equation (2.3.26).

$R_{0,n}(z) = \lambda_1^n$ implies $R_{0,n}'(1) = 0$.

$R_{0,n-1}(z) = \lambda_1^{n-1} u_n(z)$ implies $R_{0,n-1}'(1) = \lambda_1^{n-1} u_n'(1)$

and by equation (2.3.26):

$$r_j(z) = \frac{u_{j+1}(z)}{\lambda_1} r_{j+1}(z) - \frac{(j+2)\mu_1}{\lambda_1} r_{j+2}(z)$$

$$R_{0,j}'(z) = \frac{u_{j+1}'(z)}{\lambda_1} R_{0,j+1}(z) - \frac{u_{j+1}(z)}{\lambda_1} R_{0,j+1}'(z) - \frac{(j+2)\mu_1}{\lambda_1} R_{0,j+2}(z)$$

and

$$R_{0,j}'(1) = \frac{-\lambda_2 + d_j\mu_2}{\lambda_1} r_{j+1}(1) - \frac{\lambda_1 + (j+1)\mu_1}{\lambda_1} R_{0,j+1}'(1) - \frac{(j+2)\mu_1}{\lambda_1} r_{j+2}(1) \qquad 2.3.38$$

With equations (2.3.36) and (2.3.37), $\pi_i'(1)$ can be solved by the (n+1) linearly independent simultaneous equations. Numerical results are shown in the Tables 2.1, 2.2 and 2.3 and are plotted as curves of exact solution in Figures 2.6.1, 2.6.2, and 2.6.3 of Section 2.6. $\pi_i'(1)$ increases rapidly with respect to i, the number of Class I jobs in the switch. $\pi_i'(1)$ is also a function of μ_2/μ_1, the ratio of Class I service requests to that of Class II jobs. $\pi_i'(1)$ increases very rapidly with respect to μ_2/μ_1, even with fixed traffic intensity of Class I jobs and Class II jobs, i.e., λ_1/μ_1 and λ_2/μ_2 remain unchanged. The difficulty in calculating numeric results from the above algorithm will be discussed in Section 2.4, and the numerical results will be discussed in Section 2.6.

2.3.3. Multi-Server Model

If the number of servers is more than one, and a job can only be serviced by at most one server simultaneously, then some modifications have to be made for the above algorithm. Let us use the following notation:

- p = number of servers in the switch.

  c_p = channel capacity, or service ability of each server.

then $c = p * c_p$.

$d_i(j)$ = channel capacity for Class II jobs when the system is in state (i,j)

$= \min\{(c-ic_v), jc_p\}$

For all $j > (c-ic_v)/c_p$, $d_i(j)$ equals $(c-ic_v)$, the channel capacity left for Class II jobs when the system has i Class I jobs. By a straightforward manipulation, an expression similar to equation (2.3.17) can be written:

$$A(z)\pi(z) = (1-\frac{1}{z})b(z) \qquad 2.3.39$$

Instead of being constant, b(z) is now a function of z with jth element $b_j(z)$:

$$b_j(z) = \sum_{0\le j<(p-i/\beta)} (d_i-jc_p)\mu_2 z^j P_{i,j} \;/\; P_i \qquad 2.3.40$$

where $\beta=c_p/c_v$, and

$$b(z) = [b_0(z)\ b_1(z)\ b_2(z) \cdot \cdot \cdot \cdot \cdot\ b_n(z)]^T$$

Following exactly the same procedure, we derive n homogeneous equations similar to equations (2.3.24).

$$r(z_i) \cdot b(z_i) = 0 \qquad \text{for } 0<z_i<1,\ i=1,2,3,\ldots,n.$$

for all roots z_i of |A(z)| in the range (0,1), and one non-homogeneous equation similar to equation (2.3.25):

$$r(1) \cdot b(1) = \frac{d}{dz}\,|A(z)|_{z=1}$$

All of P_{ij}, such that $0\le j<(c-ic_v)/c_p$, are unknown variables in b(z), and the

number of variables exceeds the number of equations. Now some of the equations of (2.3.1) which are not used in finding equation (2.3.39) can be used as supplementary equations. These equations are:

$$[\lambda_1 + i\mu_1 + \lambda_2]P_{i,0} = \lambda_1 P_{i-1,0} + (i+1)\mu_1 P_{i+1,0} + d_i\mu_2 P_{i,1} \qquad 2.3.41$$

for all $(c-ic_v)/c_p>1$. And

$$[\lambda_1 + i\mu_1 + \lambda_2 + d_i\mu_2]P_{ij}$$
$$= \lambda_1 P_{i-1,j} + (i+1)\mu_1 P_{i+1,j} + \lambda_2 P_{i,j-1} + d_i\mu_2 P_{i,j+1} \qquad 2.3.42$$

for all $1 < j+1 < (c-ic_v)/c_p$.

For each P_{ij}, $j>0$, in $b(z)$, there will be a corresponding equation in either of equation (2.3.41) or equation (2.3.42), so the total number of equations will always equal the number of unknowns. For a special case, $c_v=c_p$, the problem is the same as [Bha75], and the number of simultaneous equations becomes $p(p+1)/2$. All P_{ij}, with $0 \le j < (c-ic_v)/c_p$, can then be solved by these equations. With all $P_{i,j}$, $0\le j<(c-ic_v)/c_p$, $b(z)$ can be calculated from equation (2.3.40). Following the procedure of the uni-server example of the last section, all system parameters can then be derived.

Next comes the example to solve $\pi_i'(1)$ for the general multi-server case.

The corresponding equation (2.3.32) becomes:

$$A'(z)\pi(z) + A(z)\pi'(z) = b(z)/z^2 + (1-\frac{1}{z})b'(z) \qquad 2.3.43$$

or

$$A(1)\pi'(1) = b(1) - A'(1)\cdot 1$$

These are the same n equations (2.3.33) as those for the uni-server case. The extra non-homogeneous equation is derived in a manner similar to that of the uni-server case.

$$\pi'(z) = \frac{R(z)}{|A(z)|}[b(z)/z^2 + (1-\frac{1}{z})b'(z) - A'(z)\pi(z)] \qquad 2.3.44$$

Let z approach to one and use the L'Hospital's Rule to calculate the limit.

$$\pi'(1)=1/[\frac{d}{dz}|A(z)|_{z=1}]\cdot\{R'(z)[b(z)/z^2+(1-\frac{1}{z}b'(z)-A'(z)\pi(z)]$$

$$+R(z)\cdot[-2b(z)/z^3+2b'(z)/z^2-A''(z)\pi(z)-A'(z)\pi'(z)]\}_{z=1} \qquad 2.3.45$$

Choose the first row, i=0, as the extra equation needed to solve $\pi'(1)$ of equation (2.3.43). The same form of equation (2.3.37) is derived:

$$\pi_0'(1) = \phi_0 + [a_2 - \sum_{j=0}^{n} \gamma_j \pi_j'(1)]/a_1 \qquad 2.3.46$$

with minor modification of a_2, where ϕ_0, γ_j and a_1 have the same definition.

$$a_2 = \sum_{j=0}^{n} r_j(1)[-2b_j(1) + 2b_j'(1) + 2d_j\mu_2]$$

with

$$\phi_0 = \sum_{j=0}^{n} R_{0j}'(1)[b_j(1) + \lambda_2 - d_j\mu_2] / a_1$$

$$\gamma_j = r_j(1)[-\lambda_2 + d_j\mu_2] = n!\,\mu_1^n \cdot (-\lambda_2 + d_j\mu_2) / (j!\,\mu_1^j)$$

$$a_1 = \sum_{j=0}^{n} \gamma_j = \sum_{j=0}^{n} r_j(1)[-\lambda_2 + d_j\mu_2]$$

$$= n!\,\mu_1^n \sum_{j=0}^{n} \lambda_1^j \cdot (-\lambda_2 + d_j\mu_2) / (j!\,\mu_1^j)$$

$R_{0j}'(1)$ are exactly the same as the uni-server case.

With equations (2.3.45) and (2.3.46), $\pi_i'(1)$ can be solved by the (n+1) simultaneous equations. Numerical results are shown in Table 2.1 and plotted in Figure 2.6.1 of Section 2.6. In both the uni-server and the multi-server systems, we can see little little variation in $\pi_i'(1)$ in over loaded states. This is a similar situation with that of average queue length of an M/M/p queue and of an M/M/1 queue with the same traffic intensity varies much less in heavy traffic than in the light traffic.

The conditional average queue length, $\pi_i'(1)$, increases rapidly with respect to i for both uni-server and multi-server. The rate of increase depends on the ratio of job requests, μ_2/μ_1. For large μ_2/μ_1, $\pi_i'(1)$ in overloaded states may be several orders of

magnitude higher than $\pi_i'(1)$ in underloaded states. Although the probability that the system will be in the overloaded states is small, the long queue length built up in these states contributes much to the total average waiting length. The long queue length in overloaded states also causes many other problems, such as the problem of buffer space for Class II jobs, flow control problems, and congestion problems.

In underloaded states differences between multi-server and uni-server will not affect the overall performance of the integrated switch as much as they will in overloaded states. For the above arguments, a multi-server integrated switch has approximately the same performance as a uni-server switch with the same total service ability.

2.4. Error Analysis and Conditional Mean Approximation

As it is mentioned in Section 2.1 (λ_1, μ_1) and (λ_2, μ_2) differ from each other by several orders of magnitude. To deal with numbers which are combinations of very large numbers and very small numbers, a relatively long computer word size should be used to retain the information. For example, if a=100, and b=0.01, then a+b=100.01 needs five digits of precision, and $(a+b)^2-a^2-2*ab$ = 10002.0001-10000-2 = 0.0001 needs as many as nine digits of precision to get the right result. A much greater need for precision arises in the algorithm of Section 2.3, in which there is high order multiplication of the form (a+b). Thus in solving the integrated switch model, the rounding error does restrict the maximum dimension which can be solved by a specific computer word size.

2.4.1. Error Analysis

Let us first review the whole algorithm. There are four steps: Find the roots of $|A(z)|$; calculate $r_j(z_i)$ which is the cofactor of $a_{j,0}(z_i)$; invert the matrix $R(z_i)$ to solve b_i; finally, calculate the required parameters from b_i's. The first step is basically calculating the determinant of a band matrix, or calculating a three-term recurrence formula. Define A_i the ith principle minor of matrix A, that is:

$$A_i = \begin{vmatrix} a_{11} & a_{12} & 0 & \cdots & 0 \\ a_{21} & a_{22} & a_{23} & \cdots & 0 \\ 0 & a_{32} & a_{33} & \cdots & a_{i-1i} \\ 0 & 0 & \cdots & a_{ii-1} & a_{ii} \end{vmatrix}$$

or

$$A_i = a_{ii} A_{i-1} - a_{ii-1} \cdot a_{i-1i} A_{i-2} \qquad i>2 \qquad 2.4.1$$

where $A_1 = a_{11}$, and $A_0 = 1$.

In general the three-term recurrence formula is not numerically stable. The above equation can be simplified to equation (2.3.26) as follows:

$$r_{j-1}(z) = \frac{u_j(z)}{\lambda_1} r_j(z) - \frac{(j+1)\mu_1}{\lambda_1} r_{j+1}(z) \qquad 2.3.26$$

where $u_j(z) = \lambda_1 + j\mu_1 + \lambda_2(1-z) + (c-j*c_v)\mu_2(1-1/z)$.

To study the numerical stability of the above equation, let us suppose that there is a rounding error, dr_j, of $r_j(z)$, then the rounding error of r_{j-1} will be

$$dr_{j-1} = [u_j(z)/\lambda_1] \cdot dr_j + [u_j'(z)/\lambda_1] \cdot r_j(z)\, dz \qquad 2.4.2$$

In general, $u_j(z)$ and $u_j'(z)$ will not be zero for z_i a root of $|A(z)|=0$, and the error of dr_j, dz_i will be exaggerated by factors $u_j(z_i)/\lambda_1$ and $u_j'(z_i)$, respectively. For a problem with $\lambda_2/\lambda_1=10^3$, for each recurrence, the rounding error will increase by the same order, or 10 bits of precision will be lost. A PDP-10 double precision word has 72 bits, and it can only solve a problem in the order of $\lambda_2/\lambda_1=10^3$ and n=7.

The above is a very rough error analysis. Consider the example:

Let $\lambda_1 = 1$, $\mu_1 = 1$, $c = 1.2$, $c_v = 1$, $\lambda_2 = 5000$, $\mu_2 = 10000$.

Then

$$u_0(z) = \lambda_1 + \lambda_2(1-z) + c\mu_2(1-1/z) = -5000z + 17001 - 12000/z.$$

$$u_1(z) = \mu_1 + \lambda_2(1-z) + (c-c_v)\mu_2(1-1/z) = -5000z + 7001 - 2000/z.$$

$$|A(z)| = u_0(z) \cdot u_1(z) - \lambda_1\mu_1$$

$$= 10^7*[2.5z^2 - 12.001z + 18.9024 - 11.8014/z + 2.4/z^2]$$

$$= 10^7*(1-1/z)\,[2.5z^2 - 9.501z + 9.4014 - 2.4/z]$$

It is easy to see that only one root, z_1, of $|A(z)|$ is in (0,1). The corresponding equation is as follow:

$$u_1(z_1)b_0 + \lambda_1 b_1 = 0$$

with z=1

$$u_1(1)b_0 + \lambda_1 b_1 = \frac{|A(z)|}{(1-1/z)}\Big|_{z=1} = 4000$$

This implies

$$b_0 = 4000 / [1-u_1(z_1)] \text{ and } b_1 = -b_0 u_1(z_1)$$

Now we have:

$$\gamma_0 = \mu_1[-\lambda_2 + c\mu_2] = 7000$$

$$\gamma_1 = -\lambda_2 + (c-c_v)\mu_2 = -3000$$

$$a_1 = \gamma_0 + \gamma_1 = 4000$$

$$a_2 = [-2b_0 + 2c\mu_2 - 2b_1 + 2(c-c_v)\mu_2] = 2*10^4$$

$$R_{01}'(1) = 0, \text{ and } R_{00}'(1) = u_1'(1) = -3000.$$

$$\phi_0 = -3000[b_0 + \lambda_2 - c\mu_2]/a_1 = 1\frac{3}{4}[b_0 - 7000]$$

The two equation of $\pi_0'(1)$ and $\pi_1'(1)$ are

$$\pi_0'(1) - \pi_1'(1) = b_0 - 7000$$

$$11\pi_0'(1) - 3\pi_1'(1) = 20 - 3[b_0 - 7000] \qquad 2.4.3$$

They can be solved as:

$$\pi_0'(1) = 2.5 - \frac{3}{4}(b_0 - 7000)$$

$$\pi_1'(1) = 2.5 - \frac{4}{7}(b_0 - 7000) \qquad 2.4.4$$

With the above formula, $\pi_0'(1)$ and $\pi_1'(1)$ are solved without any rounding error.

If there is a rounding error of z_1, then the rounding error of $\pi_0'(1)$ will be:

$$d\pi_0'(1) = -\frac{3}{4}db_0$$

$$= -3000 / [1-u_1(z_1)]^2 \cdot du_1(z_1)$$

$$= -3000 / [1-u_1(z_1)]^2 \cdot u_1'(z_1)\, dz_1$$

$$\sim -3000*7506.45\, dz_1$$

$$\sim 2.25*10^6\, dz_1 \qquad 2.4.5$$

where z_1=0.399866733, $u_1(z_1)$=$-6*10^{-5}$ and $u_1'(z_1)$=7506.45

A rounding error of dz_1 is exaggerated by a factor of 2 million, i.e., 20 bits of precision are lost to $\pi_0'(1)$.

The above example also shows that the numerical instability comes from the problem rather than the algorithm. Whatever the algorithm is, the final representative must be the same as equation (2.4.5). Hence, we should use as many bits of precision as are available to calculate the algorithm. Because of the difficulty of calculating the exact solution, a simple conditional mean approximation is suggested in Section 2.4.2, and a more complex diffusion approximation is derived in Section 2.5. The approximation will have less precision, but it can be useful when the calculation of the exact solution is not feasible.

2.4.2. Conditional Mean Approximation

Here we give a simple approximation algorithm for a uni-server integrated switch. $\pi_i(z)$ is defined as equation (2.3.11), the generating function of the number of Class II jobs in the switch, given that there are i Class I jobs in the switch. $\pi_i'(1)$, which is the average number of Class II jobs in the switch, given that there are i Class I jobs in the switch, is called the conditional mean. In the following approximation, only the parameters $\pi_i'(1)$ are estimated. Let us state the basic system equation (2.3.12) as follows:

$$[\lambda_1+i\mu_1+\lambda_2(1-z)+d_i\mu_2(1-1/z)]\pi_i(z) - i\mu_1\pi_{i-1}(z) - \lambda_1\pi_{i+1}(z)$$
$$= d_i\mu_2(1-\frac{1}{z})P_{i,0} \;/\; P_i. \qquad 2.3.12$$

Differentiating the above equation with respect to z and letting z equal 1, we will get

$$(-\lambda_2+d_i\mu_2)\pi_i(1)+(\lambda_1+i\mu_1)\pi_i'(1)-i\mu_1\pi_{i-1}'(1)-\lambda_1\pi_{i+1}'(1)=d_i\mu_2P_{i,0}/P_i \qquad 2.4.6$$

$\pi_i(1)$ equals 1, because $\pi_i(z)$ is a generating function of a distribution. The above equation can be simplified as:

$$-(\lambda_1+i\mu_1)\pi_i'(1)+i\mu_1\pi_{i-1}'(1)+\lambda_1\pi_{i+1}'(1)=-\lambda_2+d_i\mu_2(1-P_{i,0}/P_i) \qquad 2.4.7$$

In Section 2.3, we tried to solve $P_{i,0}/P_i$ and $\pi_i'(1)$. However, in this approximation we only estimate the relationship between them. In an ordinary M/M/1 queue, the average queue length $w=\rho/(1-\rho)=(1-P_0)/P_0$, or in another form

$$P_0 = 1/(1+w) \qquad 2.4.8$$

Assuming that this relation is a good approximation for the integrated switch, or

$$P_{i,0}/P_i=1/(1+\pi_i'(1))$$

we will get the following equation:

$$-(\lambda_1+i\mu_1)\pi_i'(1)+i\mu_1\pi_{i-1}'(1)+\lambda_1\pi_{i+1}'(1)=-\lambda_2+d_i\mu_2[1-1/(1+\pi_i'(1))] \qquad 2.4.9$$

for i=0, 1, 2, ·, ·, n. There are (n+1) equations and (n+1) variables, $\pi_i'(1)$ can be solved. Since, these (n+1) equations are nonlinear, an iterative method is used to solve them. Let $g_i^{(m)}$ be the mth iterative value of $\pi_i'(1)$; equation (2.4.9) can be rewritten as:

$$-[\lambda_1+i\mu_1+d_i\mu_2/(1+g_i^{(m-1)})]g_i^{(m)}+i\mu_1 g_{i-1}^{(m)}+\lambda_1 g_{i+1}^{(m)} = -\lambda_2 \qquad 2.4.10$$

With a reasonable initial guess of $g_i[(0)]$, we can solve the (n+1) simultaneous equations and get $g_i^{(1)}$ and so on, i.e., $g_i^{(2)}$, ·, ·, $g_i[(m)]$. This iterative method converges quite rapidly. Figures 2.6.1 and 2.6.2 compare the approximation results and the exact solution.

This approximation shows rapid increasing of $\pi_i'(1)$ with respect to μ_2/μ_1 as well as rapid increasing of i, as in the Kummerle's approximation, but it is much smoother and closer to the exact solution. The Bhat's algorithm and approximation are based on the multi-server system. As shown in Figure 2.6.4 of Section 2.6, the solution of

Bhat's algorithm matches other methods, but its approximation fails to show the increasing of the mean waiting length with respect to μ_2/μ_1. Unfortunately, μ_2/μ_1 in the order of thousands is the practical situation and the most important case, and the approximation differs many orders of magnitude from the exact solution.

2.5. Diffusion Approximation

If the service ability for Class II job is always greater than the input rate, $c_2\mu_2=(c-i*c_v)\mu_2>\lambda_2$, then the system will almost always have a short waiting line. If, on the other hand, there is some i such that $(c-i*c_v)\mu_2<\lambda_2$, then there will be some periods during which there will be a long queue. During time periods when input rate is greater than the service rate, a queue will be built up. The way the queue grows is a time-dependent queueing problem. It is well-known that a heavily loaded queueing system can be approximated by diffusion process [Gav68] for both asymptotic and time-dependent cases. First we will show that the solution of the queueing system in Section 2.3 is equivalent to the solution of a set of a time-dependent M/M/1 queueing system, then we will use diffusion process to approximate the overloaded states.

2.5.1. Time-Dependent System

Let an M/M/1 queueing system with Poisson input rate λ and exponentially distributed service time of mean $1/\mu$ start at time zero with initial condition h, where h is a row vector with ith element, h_i, the probability that there are i customers in the system. Suppose that the queue is finite and that $\sum h_i=1$. Distribution of the number of customers in the system at time t, p(t), satisfies the Kolmogorov forward equations[Kar66].

$$p'(t) = p(t)\,A \qquad 2.5.1$$

$$p(0) = h.$$

where p(t) is a row vector with j th element, the probability that there are j customers in the system at time t, and A is the infinitesimal generator of the system with element $a_{ij}=0$ except for $i-1\le j\le i+1$:

$$a_{i,i-1} = \mu,$$

$$a_{i,i+1} = \lambda,$$

$$a_{i,i} = -(a_{i,i-1} + a_{i,i+1})$$

$$-a_{0,0} = a_{0,1} = \lambda.$$

$$A = \begin{pmatrix} -\lambda & \lambda & 0 & 0 & & \\ \mu & -(\mu+\lambda) & \lambda & & & \\ 0 & 2\mu & -(2\mu+\lambda) & \lambda & & \\ 0 & 0 & 3\mu & -(3\mu+\lambda) & \lambda & \\ & & & \ddots & \ddots & \ddots \\ & & & i\mu & -(i\mu+\lambda) & \lambda \\ & & & & \ddots & \ddots \end{pmatrix}$$

p(t) can be solved as:

$$p(t) = h\, e^{At} \qquad 2.5.2$$

If the period t ends randomly with exponential distribution of mean 1/s, then the distribution of number of customers in the system when the period ends is:

$$p^*(s) = \int_0^\infty p(t)\, s\, e^{-st}\, dt = s\int_0^\infty h\, e^{At}\, e^{-st}\, dt$$

$$= s\, h\, [sI - A]^{-1}$$

$$= h\, [I - A/s]^{-1} \qquad 2.5.3$$

where I is the unit matrix.

Because of the extremely long holding time of Class I jobs in integrated switch, the service of Class II jobs can be modeled as one M/M/1 queue followed by another with different service rate. The service rate for Class II jobs is $d_i=(c-ic_v)$, which depends on i, the number of Class I jobs. We will model the integrated switch during

the period in which the number of Class I jobs remains the same as a simple M/M/1 queue. Those periods are of finite length, and the final distribution is expressed in terms of the initial distribution. Supposing that during period i there are i Class I jobs in the switch, we define h(i) as the initial distribution and g(i) as the final distribution. Both g(i) and h(i) are probability vectors with the jth element being the probability that the system has j Class II jobs. Either a new Class I job coming in or one of the i Class I jobs finishing its service request will cause the end of period i, so the duration of period i will be exponentially distributed with parameter $(\lambda_1+i\mu_1)$. Replacing s by $(\lambda_1+i\mu_1)$ and g(i) by $p^*(s)$, we can write equation (2.5.3) as follows:

$$g(i) = (\lambda_1+i\mu_1)\, h(i)\, [(\lambda_1+i\mu_1)I - A(i)]^{-1} \qquad 2.5.4$$

where A(i) is the infinitesimal generator of Class II jobs with i Class I jobs in the integrated switch.

$$A(i) = \begin{bmatrix} -\lambda_2 & \lambda_2 & & & & \\ d_i\mu_2 & -(d_i\mu_2+\lambda_2) & \lambda_2 & & & \\ & d_i\mu_2 & -(d_i\mu_2+\lambda_2) & \lambda_2 & & \\ & & \ddots & \ddots & \ddots & \\ & & & d_i\mu_2 & -(d_i\mu_2+\lambda_2) & \lambda_2 \\ & & & & \ddots & \ddots \end{bmatrix}$$

where $d_i=(c-i*c_v)$. Equation (2.5.4) can also be written as:

$$h(i) = g(i)\, [I - A(i)/(\lambda_1+i\mu_1)] \qquad 2.5.5$$

The beginning of one period is the end of another period of course. Thus h(i) must be equal to either g(i-1) or g(i+1), the end of a period (i-1) or a period (i+1). The transition rate from period (i-1) ,where there are (i-1) Class I jobs, is λ. The transition rate from period (i+1), where there are (i+1) Class I jobs, is $(i+1)\mu_2$. If we multiply

these transition rates by their steady state probabilities, P_{i-1} and P_{i+1} respectively, we get the following ratio of absolute transition rate:

$$h(i) = \frac{\lambda_1 P_{i-1}}{\lambda_1 P_{i-1} + (i+1)\mu_1 P_{i+1}} g(i-1) + \frac{(i+1)\mu_1 P_{i+1}}{\lambda_1 P_{i-1} + (i+1)\mu_1 P_{i+1}} g(i+1) \qquad 2.5.6$$

where P_i is the steady state probability that the system has i Class I jobs. Then

$$h(i) = \frac{i\mu_1}{i\mu_1 + \lambda_1} g(i-1) + \frac{\lambda_1}{i\mu_1 + \lambda_1} g(i+1) \qquad 2.5.7$$

Substituting equation (2.5.7) in equation (2.5.4) yields

$$g(i)\,[1 - A(i)/(i\mu_1 + \lambda_1)] = [i\mu_1\, g(i-1) + \lambda_1\, g(i+1)]/(i\mu_1 + \lambda_1) \qquad 2.5.8$$

The jth element of the above row vector is

$$g_j(i) - [g_{j-1}(i)\lambda_2 + c_2(i+1)\mu_2 g_{j+1}(i) - (\lambda_2 + c_2(i)\mu_2) g_j(i)] \,/\, (\lambda_1 + i\mu_1)$$
$$= (i\mu_1 g_j(i-1) + \lambda_1 g_j(i+1)) \,/\, (i\mu_1 + \lambda_1) \qquad 2.5.9$$

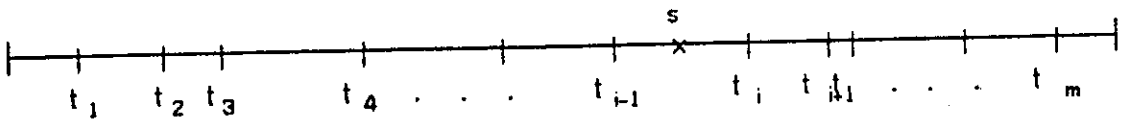
or

$$[\lambda_1 + i\mu_1 + \lambda_2 + c_2(i)\mu_2]\, g_j(i)$$
$$= \lambda_1 g_j(i+1) + i\mu_1 g_j(i-1) + \lambda_2 g_{j-1}(i) + c_2(i)\mu_2 g_{j+1}(i) \qquad 2.5.10$$

where $c_2(i) = c - i*c_v$.

This is exactly the same balance equation set (2.3.1) of Section 2.3, with P_{ij}/P_i as $g_j(i)$. Thus the integrated switch model is exactly a set of time-dependent simple queueing systems. Notice that $g_j(i)$ is the distribution of Class II jobs when the system leaves period i, while P_{ij}/P_i, defined in Section 2.3, is the steady state distribution of Class II jobs when the system is in i Class I state. The reason that they are the same can be explained by the "Waiting Time Paradox". This paradox arises from random sampling of Poisson process:

$(t_1, t_2, t_3, \ldots\ldots, t_m)$ are the times when the events of the Poisson process happen, and s is a random sample point. Suppose that s falls in (t_{i-1}, t_i).



The paradox is:

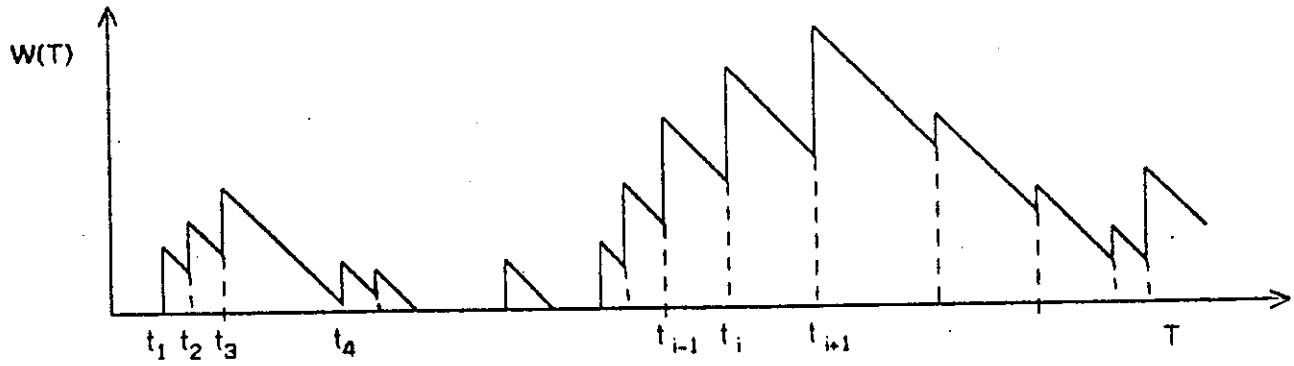
The length of periods $(s-t_{i-1})$ and (t_i-s) are of the same exponential distribution as (t_j-t_{j-1}) for all $j\neq i$.

The paradox says that the distribution of $(s-t_{i-1})$ and (t_i-s) is exactly the same as the normal interarrival time of this Poisson process rather than half of it. The reason is that the random sampling itself can be treated as another event of this Poisson process. This is exactly the same as the relationship between the integrated switch model and the time-dependent model. The steady state distribution $P_{i,j}/P_i$ is sampled randomly between two transitions, and the sample point itself is an event of the Poisson process which determines when the next next Class I transition will happen.

2.5.2. Simple Diffusion Model

First we will analyze a simple diffusion approximation model for an M/M/1 waiting system. Consider a single servicing facility at which customers arrive in a Poisson fashion with rate λ. The service times S, are independent random variables with exponential distribution of mean $1/\mu$. Let W(t) represent the (virtual) waiting time at t, i.e., the time a customer arriving at t waits in queue. Then W(t), considered as a function of time, is a spatially homogeneous random process, modified by a reflecting barrier at W=0. An actual W(t)-path is a random sawtooth, exhibiting vertical jumps of

service-time magnitude at the instants of customers arrival and, otherwise, diminishing deterministically at slope -1 until the barrier at zero is reached.



If the traffic intensity parameter $\rho=\lambda/\mu$ is close to unity, then it is clear that W(t) is seldom near the barrier and is typically large with respect to changes in W(t) likely to occur in small time intervals. Thus it becomes plausible to replace W(t), t>0, with an approximating continuous, diffusion process. For mathematical details concerning such a model see Kingman [Kin64] and Gaver [Gav68].

To approximate the distribution of W(t), compute the infinitesimal mean (drift), b, and variance, a , from

$$bh \sim E[W(t+h) - W(t) \mid W(t)\ ] = \{\lambda E[S] - 1\}h + 0(h) = (\rho-1)h + 0(h) \qquad 2.5.11$$

and

$$ah \sim Var[W(t+h) - W(t) \mid W(t)] = \lambda E[S^2]h + 0(h) = (2\lambda/\mu^2)h + 0(h) \qquad 2.5.12$$

and then solve the forward differential (Fokker-Planck) equation for the distribution function F(x,t; w)

$$\frac{\partial F}{\partial t} = -b\,\frac{\partial F}{\partial x} + \frac{a}{2}\,\frac{\partial^2 F}{\partial x^2}$$

$$= -(\rho-1)\,\frac{\partial F}{\partial x} + \frac{\lambda}{\mu^2}\,\frac{\partial^2 F}{\partial x^2}$$

$$= -(\rho-1)\frac{\partial F}{\partial x} + \frac{\rho^2}{\lambda}\frac{\partial^2 F}{\partial x^2} \qquad 2.5.13$$

subject to the initial condition

$$F(x,0;\, w) = \begin{cases} 1 & x \geq w \\ 0 & x < w \end{cases}$$

and the boundary condition

$$F(x,t;\, w) = 0 \qquad (x<0,\ t\geq 0)$$

$\{W_d(t),\ t\geq 0\}$ denotes the diffusion process whose transition probabilities are specified by the above equations. Heavy traffic theory suggests that if the parameter

$$\frac{1-\rho}{2\rho^2/\lambda} = \frac{-b}{a} \qquad 2.5.14$$

is positive and small, we can expect the distribution function F of $W_d(t)$, which is the solution of equation (2.5.13), to provide a good approximation to the exact distribution of W(t). If equation (2.5.14) is positive, the limiting distribution of W(t) as $t\to\infty$ exists. If equation (2.5.14) is negative or zero no such limit exists, but we are interested in approximating the distribution of W(t) for finite t and will consider the solution of our diffusion equations for this purpose.

An explicit solution to the diffusion problem defined by equation (2.5.13) is given by Chandrasikhar[Cha43]. However, for our purpose only the transform solution to the diffusion problem is needed, and we record it here:

$$\tilde{f}(\xi,s;\, w) = \int_0^\infty e^{-st} \int_0^\infty e^{-\xi x}\, dF(x,t;\, w)dt \qquad 2.5.15$$

Assume convergence for at least s, $\xi>0$. Straightforward manipulation of equation (2.5.13) yields

$$[-s + (1-\rho)\xi + \rho^2\xi^2/\lambda]\, \tilde{f}(\xi,s;\, w) = c_1 + \xi c_2 - \psi(\xi) \qquad 2.5.16$$

where operator $\frac{\partial}{\partial t}$ is replaced by s and $\frac{\partial}{\partial x}$ is replaced by ξ. By including the particular integral to the right-hand side, we can rewrite the equation as:

$$\tilde{f}(\xi,s;\ w)= \left[\frac{c_1+\xi c_2-\psi(\xi)}{(\rho^2/\lambda)\xi^2+(1-\rho)\xi-s}\right] \qquad 2.5.17$$

where c_1 and c_2 depend only upon s, and

$$\psi(\xi) = \int_0^\infty e^{-\xi x}\, dF(x,0;\ w) \qquad 2.5.18$$

the transformation of $F_w(x)$ represents the distribution function of the initial condition w.

The denominator has the two real zeros

$$\xi_i = \frac{\lambda(\rho-1)}{2\rho^2}\left[1 \pm \left(1 + \frac{4s\rho^2}{\lambda(1-\rho)^2}\right)^{1/2}\right] \qquad i=1,2 \qquad 2.5.19$$

The signs of ξ_i, i=1,2 depend upon the traffic intensity $\rho=\lambda/\mu$; if $\rho<1(>1)$ then $b<0(>0)$, and $\xi_2>0(\xi_1>0)$. We are only interested in $\rho>1$ case, so we define

$$\xi_1 = \frac{\lambda(\rho-1)}{2\rho^2}\left[1 + \left(1 + \frac{4s\rho^2}{\lambda(1-\rho)^2}\right)^{1/2}\right] \qquad 2.5.20$$

We can cancel the pole by putting in

$$c_1+c_2\xi_1 = \psi(\xi_1) \qquad 2.5.21$$

The condition that $s\tilde{f}(0,s;\ w)=1$ has provided the information that $c_1=1$, so equation (2.5.17) becomes:

$$\tilde{f}(\xi,s;\ w) = \frac{\lambda}{\rho^2}\,\frac{\xi\xi_1^{-1}\psi(\xi_1)-\psi(\xi)}{(\xi-\xi_1)(\xi-\xi_2)} \qquad 2.5.22$$

Several simple interpretations of and comments on these results now follow.

Remark 1 Time-dependent waiting time behavior is reflected in behavior of the mean waiting time: E[W(t) | W(0)=w]. The time transformation of E[W(t) | W(0)=w], or E[W(t) | N(0)=i], has been studied, both numerically and asymptotically, in Gaver[Gav66]. The mean and variance of the transformation of the diffusion process

are obtained by differentiating equation (2.5.22) at $\xi=0$; the result may be expressed as:

$$\int_0^\infty s\, e^{-st}\, E[W_d(t) \mid W_d(0) \text{ with distribution } \psi(w)]\, dt$$

$$= \frac{d}{d\xi} \tilde{f}(\xi,s;\, w)\,|_{\xi=0}$$

$$= \frac{-\lambda s}{\rho^2(\xi-\xi_1)(\xi-\xi_2)} \{\xi_1^{-1}\psi(\xi_1) - \psi'(\xi) - [\frac{1}{(\xi-\xi_1)} + \frac{1}{(\xi-\xi_2)}]\,[\xi\xi_1^{-1}\psi(\xi_1) - \psi(\xi)]\}_{\xi=0}$$

$$= -\psi'(0) + (\rho-1)/s + \xi_1^{-1}\psi(\xi_1)$$

$$= \bar{w} + (\rho-1)/s + \xi_1^{-1}\psi(\xi_1) \qquad 2.5.23$$

where

$\bar{w} = -\psi'(0)$ = mean value of the initial condition

$\psi(0) = \int_0^\infty dF_w(x) = 1.$

$\xi_1\xi_2 = -\lambda s/\rho^2$ and $\xi_1+\xi_2 = \lambda(\rho-1)/\rho^2.$

The mean value of an exponentially distributed observation will be a combination of three terms, the initial value, a linear increase(or decrease, depending on the value ρ; here we only consider $\rho>1$) deterministic term and a dispersion term. The third term:

$$\xi_1^{-1}\psi(\xi_1) = \int_0^\infty \xi_1^{-1}\, e^{-\xi_1 x}\, dF_w(x)$$

$$= \int_0^\infty F_w(x)\, e^{-\xi_1 x}\, dx \qquad 2.5.24$$

is bounded by $\xi_1^{-1} < \rho^2/[\lambda(\rho-1)]$ for $\rho>1$. If the mean observation time is relatively long, $E[\tau]\to\infty$, or $s\to 0$, then the second term will dominate, and will be approximate to $(\rho-1)/s$.

Remark 2 Let the expected observation time be $E[\tau]\to\infty$, or equivalently $s\to 0$. Then

$$\xi_i = \frac{\lambda(\rho-1)}{2\rho^2}\left[1 \pm \left(1 + \frac{4s\rho^2}{\lambda(1-\rho)^2}\right)^{1/2}\right] \qquad i=1,2$$

$$\sim \frac{\lambda(\rho-1)}{2\rho^2}\left[1 \pm \left(1 + \frac{2s\rho^2}{\lambda(1-\rho)^2}\right)\right] \qquad i=1,2$$

Then

$$\xi_1 \sim \lambda(\rho-1)/\rho^2 + s/(\rho-1) \qquad 2.5.25$$

$$\xi_2 \sim -s/(\rho-1) \qquad 2.5.26$$

and

$$\int_0^\infty se^{-st}\,\mathrm{Prob}[W_d(t)=0 \mid W_d(0)=w]\,dt$$

$$= \lim_{\xi\to\infty} s\xi\ \tilde{f}(\xi,s;\,w)$$

$$= (\lambda/\rho^2)\, s\xi_1^{-1}\,\psi(\xi_1)$$

$$\sim s(\rho-1)^{-1}e^{-\xi_1 w} \qquad \text{for } \rho>1$$

$$\sim O(s) \qquad \text{for } \rho>1 \text{ and } s\to 0 \qquad 2.5.27$$

Thus the probability that the system will hit the boundary, $W_d(t)=0$, decreases in the order of s. When s is small and $\rho>1$, the diffusion approximation will differ from the real process only in the order of O(s).

Remark 3 For $\rho<1$, the above approximation becomes very sensitive to the third, dispersion, term of equation (2.5.23). Because it is very hard to estimate the exact distribution of the number of customers when the number of Class I job changes, we use Gaver's[Gav66] approximation formula for a time-dependent M/M/1 queueing system instead of the technique of diffusion approximation. The formula is:

$$\int_0^\infty se^{-st}\,E[N(t) \mid N(0) = i\,]\,dt$$

$$= i + \mu(\rho-1)(1-x^i) + [1+\mu(\rho-1)(1-x)s]\cdot x^i\lambda/(s+\lambda(1-x)) \qquad 2.5.28$$

where

$$x = 2/\{1 + (s+\lambda)/\mu + [(1 + (s+\lambda)/\mu)^2 - 4\lambda/\mu]^{1/2}\}$$

From $E[W(t)] = E[N(t)]/\mu$, we can get

$$E[\, W(t) \mid W(0)=w \,] = E[\, W(t) \mid N(0)=w\mu \,] = (1/\mu)\, E[\, N(t) \mid N(0)=w\mu \,] \qquad 2.5.29$$

2.5.3. Diffusion Approximation for the Integrated Switch

There is more than one simple diffusion process for the integrated switch model. Actually the system is modeled as one simple diffusion process followed by another. Suppose that the simple diffusion process i has the following parameters and that the notations in the parenthesis used in section 2.5.2 will be replaced by the right-hand terms of the following formulas.

input rate $(\lambda) = \lambda_2$

service rate $(\mu) = (c-ic_v)\mu_2 = b_i$

traffic intensity $(\rho) = \lambda_2/[(c-ic_v)\mu_2] = \eta_i$

mean observation time for an exponentially distributed sampling (s)

$= \lambda_1 + i\mu_1 = s_i$

Definitions of h_i and g_i similar to those in section 2.5.1 are used:

h_i = average waiting time of Class II jobs in the system when the system has i Class I jobs.

g_i = average waiting of Class II jobs in the system when the system does not have i Class I jobs.

Only the time-dependent processes with $\eta_i > 1$ are modeled by the diffusion process. Thus:

positive pole of $\tilde{f}(\xi,s;\, w)$ (ξ_1)

$$= \frac{\lambda_2(\eta_i-1)}{2\eta_i^2}\left[1 + \left(1 + \frac{4s_i\eta_i^2}{\lambda_2(1-\eta_i)^2}\right)^{1/2}\right] \qquad 2.5.30$$

$= \xi_i$

For $\eta_i>1$ and $s\to 0$, the third term of equation (2.5.23) is not very sensitive to the result, as we stated before. Thus instead of estimating the distribution of every initial condition of the diffusion process, we assumed the initial condition w. Substituting these values into equation (2.5.23), we have:

$$g_i = h_i + (\eta_i-1)/s_i + \exp[-\xi_i h_i]/\xi_i \qquad 2.5.31$$

For $\eta_i<1$, a more complicated version of equation (2.5.28) is used. The initial condition is assumed to be h_i rather than a random variable with some unknown distribution. We can calculate g_i in terms of h_i.

$$g_i = h_i + d_i(\eta_i-1)(1-x_i^{h_i/d_i}) + [1+d_i(\eta_i-1)(1-x_i)s_i]x_i^{h_i/d_i}\lambda_2/(s_i+\lambda_2(1-x_i)) \qquad 2.3.32$$

Now we have (n+1) equations with 2(n+1) unknowns, h_i and g_i for i=0,1,2, . , n. As in equation (2.5.7), we can get another set of equations:

$$d_i h_i = r_i(g_{i-1}d_{i-1}) + (1-r_i)(g_{i+1}d_{i+1}) \qquad 2.5.33$$

where $d_i h_i$ or $d_i g_i$ is the average number of Class II jobs rather than the average waiting time. And

$$r_i = i\,\mu_1 / (i\,\mu + \lambda_1') \qquad 2.5.34$$

is the probability that the transience to i Class I job is due to a new incoming Class I job. The detailed derivation is the same as equation (2.5.7).

There are (n+1) equations for equation (2.5.33). With the (n+1) equations of (2.5.33) and the (n+1) equations of (2.5.31), g_i and h_i can be solved numerically. From the derivation, we will expect a better approximation if the ratio μ_2/μ_1 is large or s_i is small. In such situations, the overloaded states will last longer, and the estimation of the final value will be more insensitive to the initial condition.

2.6. Comparison of Results

Even when the ratios λ_1/μ_1, λ_2/μ_2 are constant, the average number of Class II packets increases rapidly as the ratio μ_2/μ_1 increases. This behavior shows the similarity of the integrated switch discussed in Section 2.3, 2.4 and 2.5 and the simple models discussed in Section 2.2 and Figures 2.2.7 and 2.2.8. Similar behavior is also discovered by Bhat [Bha75]. Unfortunately, however, their approximation algorithm lost an important characteristics of the system, i.e., as shown in Figure 2.6.4 the algorithm of Bhat is not sensitive to changes in the ratio of μ_2/μ_1.

Table 2.1 shows the average number of Class II packets in the system when there are i Class I jobs. For different p, the number of servers, $\pi_i'(1)$ differ very little from each other in overloaded states, in which $d_i \leq \lambda_2$. Because of this characteristics, the CM (conditional mean) approximation algorithm based on the uni-server system becomes a good approximation for multi-server system. Another very important characteristic is that the conditional mean number of packets varies greatly for different i. Even with a small total average waiting length, the probability of overflow, which is defined as the condition in which the number of packets in the system exceeds available buffer space, will be quite high.

Tables 2.2 and 2.3 provide the comparison of the results of our integrated switch model to that of Kummerle's and Fischer's, respectively. In Kummerle's approximation, the service scheme is the same , but the service time of packets is constant rather than exponentially distributed. Kummerle's approximation uses two different formulas to calculate the mean number of Class II jobs in the system for overloaded states, $\lambda_2 \geq (c-ic_v)\mu_2$, and for underloaded states, $\lambda_2 < (c-ic_v)\mu_2$. The curve

looks a little irregular at i=7 and i=8, where $\lambda_2=(c-ic_v)\mu_2$. Both Kummerle's and the CM approximation and the exact solution show a sharp increase of the conditional mean in overloaded states. In Table 2.2 we also notice that the average queue length increases with respect to μ_2/μ_1, although λ_1/μ_1 and λ_2/μ_2 remain constant. Fischer and Harris [Fis75] fail to show this characteristic. In a later report [Bha75], they provide a new model which has this characteristic, but, in that model, they only tend to give only the total mean waiting time. As we stated before, total mean waiting time is not enough to describe the whole system. The report of Bhat and Fischer [Bha75] covers the special condition of our analysis where c=p and $\mu_2=1$.

Figures 2.6.1, 2.6.2, and 2.6.3 show the numerical results of Tables 2.1 and 2.2. Figure 2.6.4 shows the results of Table 2.3.

Table 2.1 Comparison of exact solution and approximation

$c=15$; $n=10$; $\lambda_1/\mu_1=5.0$; $\lambda_2/c\mu_2=7/15$; $\mu_1/\mu_2=100$;

$p = 1, 3, 5$.

$\pi_i'(1)$	P_i	p=1	p=3	p=5	CM Approx.
i= 0	.0068315	0.8881841	1.5899381	2.4360016	.876
i= 1	.0341575	1.0399755	1.7091119	2.5267855	1.002
i= 2	.0853938	1.2894188	1.9175224	2.6971034	1.170
i= 3	.1423231	1.7669488	2.3417091	3.0699985	1.405
i= 4	.1779038	2.7981681	3.3011327	4.0220692	1.762
i= 5	.1779038	5.1520923	5.5606589	6.2629728	2.399
i= 6	.1482532	10.4236670	10.7955900	11.3819070	4.188
i= 7	.1058951	21.0693230	21.4009170	21.9430840	11.637
i= 8	.0661845	38.8442910	39.1415930	39.6471080	29.404
i= 9	.0367691	61.7339510	62.0130820	62.4995080	53.226
i=10	.0183846	81.8106690	82.0841770	82.5715340	73.894
Total mean		11.93888501	12.38936477	13.05113306	8.195

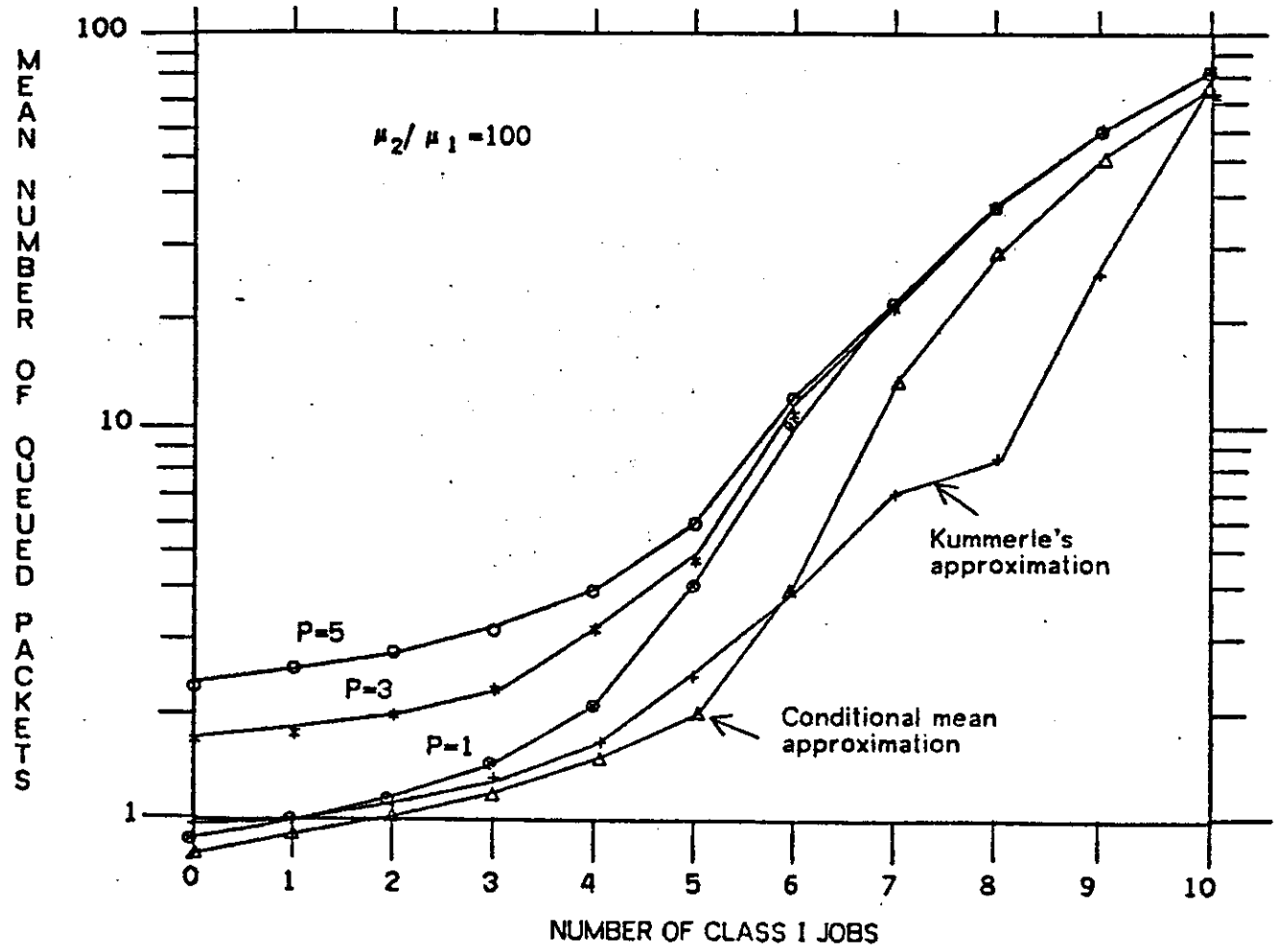


FIGURE 2.6.1 CONDITIONAL QUEUE LENGTH OF INTEGRATED SWITCH

Table 2.2 Comparsion of Kummerle's approximation

$c=15,\ p=1;\ \lambda_1/\mu_1=5.0;\ \lambda_2/c\mu_2=7/15;$

	Kummerle's Appr.		Exact Calcul.		CM Approx.		Diff. Approx.	
$\mu_2/\mu_1 =$	10	100	10	100	10	100	10	100
i:								
0	.93	.93	.92	.88	.89	.87	.84	.87
1	1.07	1.07	1.10	1.04	1.02	1.00	.94	.99
2	1.25	1.25	1.36	1.29	1.20	1.17	1.07	1.15
3	1.50	1.50	1.77	1.77	1.47	1.40	1.27	1.38
4	1.88	1.88	2.41	2.80	1.91	1.76	1.69	1.71
5	2.50	2.50	3.40	5.15	2.70	2.40	3.03	2.28
6	3.75	3.75	4.84	10.42	4.07	4.19	6.80	4.41
7	7.50	7.50	6.81	21.07	6.18	11.64	14.52	13.18
8	8.50	8.50	9.25	38.84	8.89	29.40	27.28	38.45
9	11.70	27.77	11.89	61.73	11.81	53.22	29.50	60.67
10	18.04	73.33	14.03	81.81	14.14	73.89	31.50	80.67

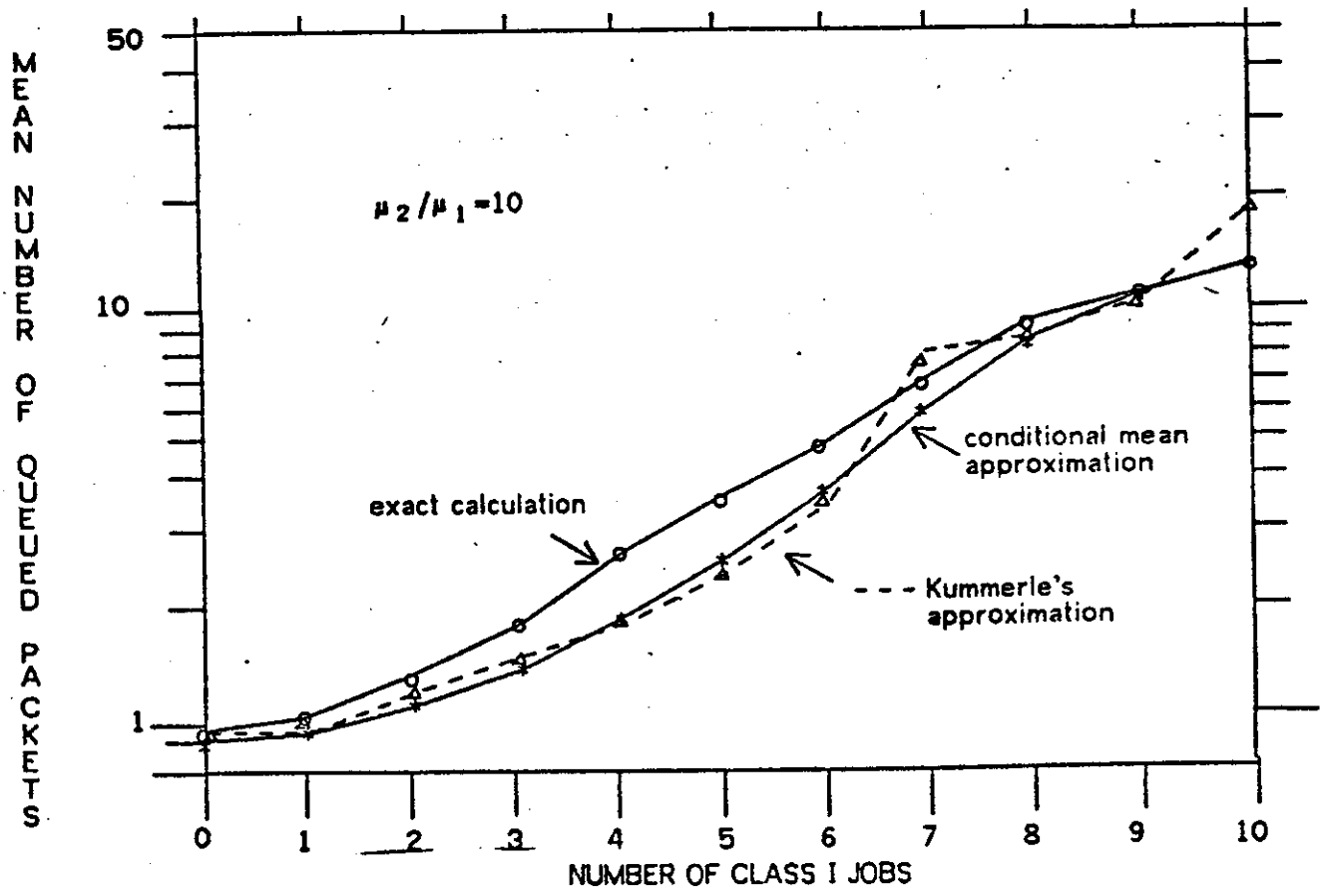


FIGURE 2.6.2 COMPARISON OF DIFFERENT APPROXIMATION

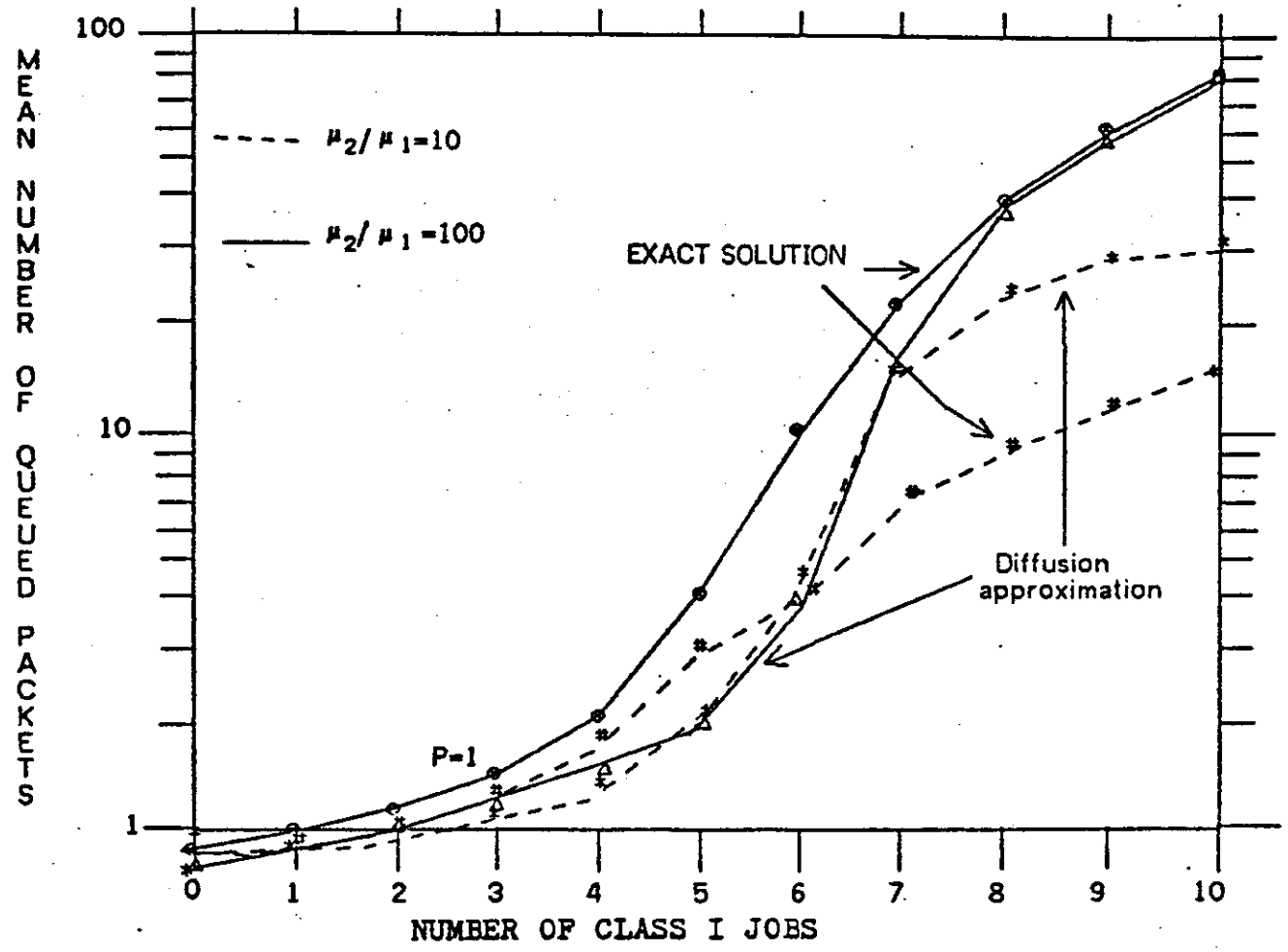


FIGURE 2.6.3 DIFFUSION APPROXIMATION

Table 2.3 Comparsion of Bhat's model

for 2 channels, ρ_1=0.5, ρ_2=0.5, $\alpha=\mu_2/\mu_1$

	Bhat's Model			Cond. Mean Approx.	
α	PB	$E[Q_2]$	$E_a[Q_2]$	PB	$E_a[Q_2]$
.0005	.2461	.5333	.5001	.1429	.3335
.005	.2461	.5336	.5005	.1429	.3352
.05	.2462	.5368	.504	.1429	.3517
.5	.248	.5618	.5505	.1429	.4867
1	.25	.5833	.5833	.1429	.6026
5	.2574	.6945	.6605	.1429	1.205
25	.2629	1.046	.6965	.1429	3.071
100	.2646	2.212	.705	.1429	8.585
500	.2651	8.34	.71	.1429	35.835
1000	.2652	15.994	.71	.1429	69.552
5000	.2652	77.219	.705	.1429	338.839

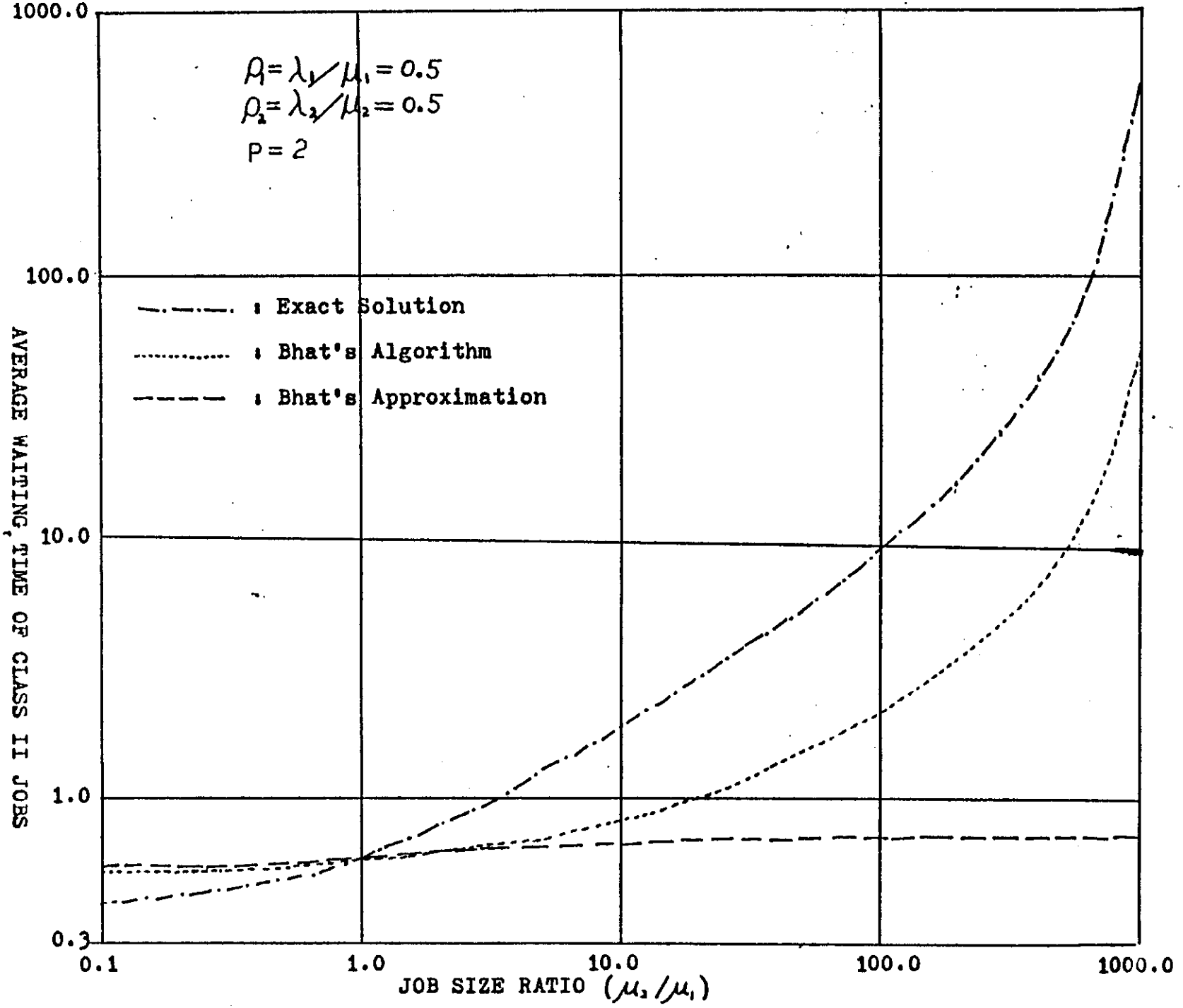


FIGURE 2.6.4 COMPARISON OF BHAT'S RESULT

CHAPTER 3 Memory Management for Data Buffers

3.1. Introduction

In Chapter 2, the waiting time for data packets was calculated. In ordinary M/M/c or M/G/c queueing systems when the waiting space is several times larger than the average queue length, the infinite queueing space assumption turns out to be good a approximation. But, in the case of the integrated switch, the variation of queue length is very large when the number of background Class I jobs varies. The queue length of Class II jobs will increase almost linearly with respect to time in some period, and the conditional mean queue length may be much longer than total mean queue length. In the under-loaded states, infinite queueing space might well be a good approximation, while in over-loaded states this assumption becomes questionable. When the number of packets requesting the service of the switch is greater than the number of packet buffers in the switch, incoming packets will suffer extra delay: They will experience a retransmission time because no empty buffer in the switch as well as the waiting time for all the packets before it to be processed. In packet-switching, an incoming packet without a buffer is not acknowledged, so this packet will be sent again. Packets requesting service of an over-loaded switch have to suffer this extra delay of re-transmission. In this chapter, finite memory space is assumed, and the mechanism and the performance of memory managements as a means of decreasing or eliminating packet delay are discussed.

There is a finite memory space M. When a packet comes in, it is assigned a buffer if one is free. There are many approaches to assigning the buffer from the

memory, and there may be a secondary storage for packets during the period of over-loaded states. Before the details of memory management are discussed, one assumption is made: The holding time for Class I job is so long that the transient of changing the number of Class I jobs is relatively short; in fact, it is negligible in terms of the whole system behavior. Thus the switch is assumed to be in one steady state followed by another with different service ability for Class II data packets.

An incoming packet is either allocated a buffer or blocked. If a buffer is assigned to a packet, that buffer is called occupied; otherwise it is free. An occupied buffer is freed after its assigned packet is processed, transmitted to the next node, and given a positive acknowledgement. This whole period is called the life-time of the buffer. If a packet can not be allocated a buffer, the switch behaves as if it has never received that packet. No acknowledgement is sent back to the sender, and by the high level control protocol, this packet will be sent by the sender again after a waiting period. Although this data packet is not lost, it does suffer an extra delay. Because of the above properties, the performance criterion used in this chapter is the probability of packets being blocked. Another criterion, memory utilization or buffer utilization, is also used in some sections.

Let us review the parameters of the integrated switch. It is a communication system for voice and data. It has very strict real time constraints. Because the system sent out a frame every frame period, say of 10 milliseconds, any memory management which needs more than a frame period to complete is not acceptable. Only simple memory management schemes can be used. Secondary storage, say a disk, will generally have access time in the order of tens of milliseconds, so a direct use of disk for input and output will be inadequate. On the other hand, if the disk is not

directly used in I/O but used, instead, in storage of the excess data packets, than it may well be effective. The access time for secondary storage may be less than the extra delay suffered by the blocked packets. According to the SENET implementataion, no processes can be overlapped when the input trunk line is writing into the input frame buffer or when the output trunk line is sending out information from the output frame buffer. Thus the extra delay for blocked packets, which is at least a two-frame period, will be in the order of 30 to 50 milliseconds. Apparently, with good use of secondary storage, fewer retransmissions will occur and the Class II data packets can have a smoother flow and a shorter delay.

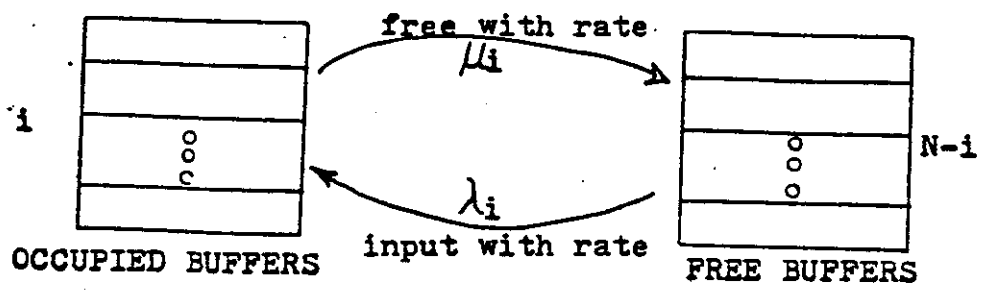
In the second section, several buffer management methods for primary memories are tried. Independent exponentially distributed random variables for the life time of buffers are assumed. The effectiveness of these managements methods are compared. In the third section, a network model is built for buffered packets flowing in the communication network. Closed queueing network model of exponential server are solved by using the algorithm of Buzen[Buze 73]. In section four, secondary storage is modeled. A forward and backward algorithm is developed to analyze the effect of the secondary storage.

3.2. Node Model

A finite memory of size M is assumed in the integrated switch. When a packet of size r comes in, how should it be assigned a buffer? Because of the real time requirements of the system, only several simple memory managements are relevant to our attempt to answer this question. These include division of memory into maximum size buffers, division of memory into several different fixed size buffers and dynamic allcoation of first-fit memory are discussed. We will let D indicate the upper bound of packet size; N=M/D, the number of buffers if all buffers are maximum size. First we will discuss the system with N buffers of size D. In this system, an incoming packet, regardless of its size, is assigned a buffer until all the buffers are occupied. Then a system with N_1 buffers of size M_1, N_2 buffers of size M_2, . , . , . , and N_k buffers of size M_k is discussed, where $M_1<M_2<\cdot<\cdot<\cdot<M_k=D$. There are two disciplines, static and dynamic, for buffer assignment when a small packet comes in while all corresponding size buffers are occupied. We can either assign a larger buffer to the packet or just block it. Finally, we explore a relatively complicated first-fit dynamic memory allocation scheme. The input stream of data packets is assumed to be Poisson and the buffer life-times are independent exponentially distributed random variables, regardless of their size.

3.2.1. Maximum Size Buffers

All the buffers are of size D, the maximum size a packet can be. A buffer in the memory is either free or occupied:



If a packet comes into a node and can not be allocated a free buffer, the packet is blocked. The probability of this happening depends on the total number of buffers, N, and the life time of buffers. With assumption of Poisson input and exponentially distributed buffer life time, a simple Markov Chain can be built to model the buffer behavior.

Let the state of the system be the number of occupied buffers. λ_i is the rate of incoming packets, and μ_i is the rate of freeing occupied buffers. The result of this model will be used later, so a general λ_i and μ_i are assumed. Both λ_i and μ_i can depend on i in some complicated form. Let P_i be the steady state probability that system is at state i. The set of balance equations is:

$$(\lambda_i+\mu_i)\, P_i = \lambda_{i-1}\, P_{i-1} + \mu_{i+1}\, P_{i+1} \qquad \text{for } i=1,2,.,.,N-1$$

$$\lambda_0\, P_0 = \mu_1\, P_1 \qquad 3.2.1$$

$$\lambda_{N-1}\, P_{N-1} = \mu_N\, P_N$$

The above equations can be simplified as

$$\lambda_i\, P_i = \mu_{i+1}\, P_{i+1} \qquad \text{for } i=0,1,2,.,.,N-1$$

Because P_i is a probability, i.e., $\sum_{i=0}^{N} P_i = 1$, P_i can be solved as:

$$P_i = P_0 \cdot \prod_{j=0}^{i-1} (\lambda_j/\mu_{j+1}) \qquad 3.2.2$$

and

$$P_0 = [1 + \sum_{i=1}^{N} \prod_{j=0}^{i-1} (\lambda_j/\mu_{j+1})\,]^{-1} \qquad 3.2.3$$

This model looks quite simple but the result is quite general. Complicated queueing models in the next section can be solved in the above form only if μ_i depends on i in a complicated way.

Non-priority System

If we assume $\mu_i = i\,\mu$, then

$$P_n = (\lambda/\mu)^n/n\,! \cdot P_0 \qquad 3.2.4$$

where

$$P_0 = [\sum_{i=0}^{n} (\lambda/\mu)^i/i\,!]^{-1}$$

The Erlang 2 formula, $E_2(n,\lambda,\mu)$, is the P_n with input rate λ, service rate μ of a M/M/n/n queueing system. Then from simple arithmetical manipulation we can get:

$$E_2(n+1,\lambda,\mu) = [1 + (n+1)\mu/\lambda \;/\; E_2(n,\lambda,\mu)\;]^{-1} \qquad 3.2.5$$

The buffer utilization factor R is defined as:

$$R = \sum_{j=1}^{n} \frac{j}{n} P_j$$

$$= \lambda/(n\mu)\,[1-E_2(n,\lambda,\mu)] \qquad 3.2.6$$

Notice that R is the utilization factor of buffers not the real utilization of the memory. The real memory utilization factor will only be half of R if the packet size is uniformly distributed between 0 and maximum size D.

Reserve Priority System

A simple reserve priority system is considered here. Suppose that there is a probability α that an incoming packet has higher priority, and that the life time of buffers assigned to a high priority packet is the same as that of other buffers. When the number of buffers occupied is more than N_1, a packet will be allocated a buffer only if it has high priority. No preemptive of buffer is allowed. For low priority packets the system has only N_1 buffers, but for high priority packets the system has N buffers. Equation 3.2.2 can be used with the following definition of λ_i and μ_i :

$$\lambda_i = \begin{cases} \lambda & 0 \le i \le N_1 \\ \alpha\lambda & N_1 < i \le N \end{cases} \qquad 3.2.7$$

$$\mu_i = i\mu$$

If P_i is the probability that the system is in state i, then the blocking probability for high priority packets is P_N, and the blocking probability for low priority packets is $\sum_{i=N1}^{N} P_i$. Figure 3.2.1 shows the effect of this priority assignment. Under a heavily loaded situation, λ/μ=80, N=30, a little extra buffer reserved for higher priority packets will decrease the blocking probability greatly, from 0.63 to 0.04 with N_1=28.

3.2.2. Different Fixed Size Buffers

Instead of dividing the whole memory into buffers of maximum size, the memory is divided into fixed size buffers of more than one size. When a data packet comes in, an attempt is made to allocate a buffer of corresponding size to that packet. If all the buffers of corresponding size are occupied, there will be two disciplines, the packet will either be blocked or assigned a larger buffer. We call the first discipline, static assignment; the second, dynamic assignment.

Static Assignment

There is a corresponding set of buffers which can be assigned to every packet. Suppose there are d different buffer sizes, each of size M_i, i=1,2, ., ., K, and $M_1<M_2<\cdot<\cdot<M_K$. There are N_i buffers of size M_i. $\sum_{i=1}^{k} M_i \cdot N_i \le M$. Any packet of size in the range $(M_{i-1}, M_i]$ is treated as a packet of size M_i and is assigned to a buffer of size M_i. Because of the simple discipline for buffer allocation, the blocking probability for packets of size M_i is independent of the allocation of buffers to packets of other sizes. Let λ_i be the Poisson input rate of packets of size M_i. With the assumption that the life time of a buffer is independent of its size, B_i, the blocking probability of packets of size M_i, is of Erlang 2 formula:

$$B_i = E_2(N_i, \lambda_i, \mu) \qquad 3.2.8$$

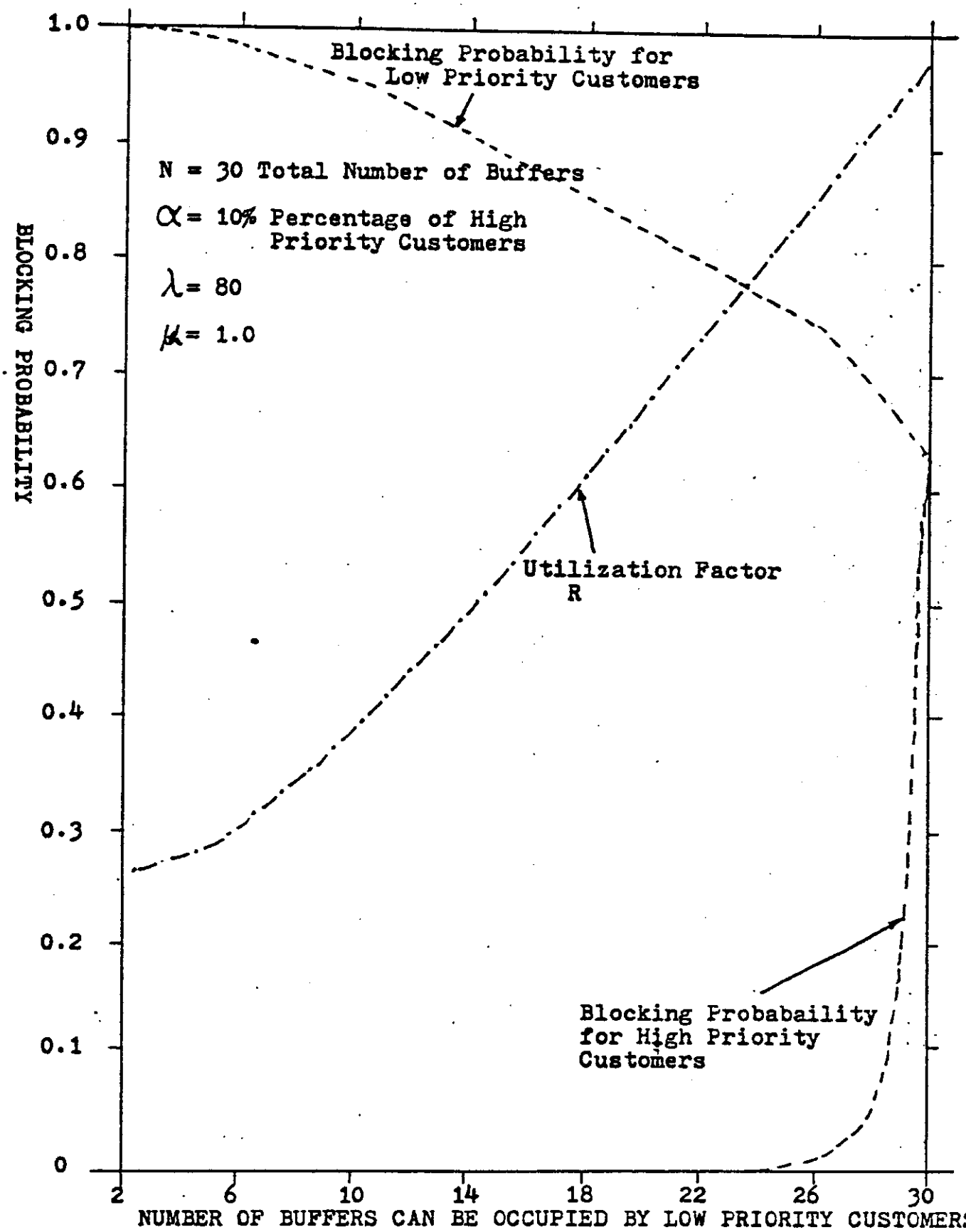


FIGURE 3.2.1 PERFORMANCE OF RESERVE PRIORITY BUFFER SYSTEM

The total blocking probability becomes:

$$B = \sum_{i=1}^{k} (\lambda_i/\lambda) \cdot B_i = \sum_{i=1}^{k} \lambda_i E_2(N_i,\lambda_i,\mu) / \lambda \qquad 3.2.9$$

where λ is the total input rate and $\lambda = \sum_{i=1}^{k} \lambda_i$.

Then R_i, the buffer utilization factor for size M_i, becomes:

$$R_i = \lambda_i[1 - E_2(N_i,\lambda_i,\mu)] / (N_i\mu) \qquad 3.2.10$$

The numerator is the effective number of buffers allocated per unit time, and the denominator is the effective number of buffers available per unit time. The total buffer utilization factor, R, is:

$$R = \sum_{i=1}^{k} M_i \cdot N_i \cdot R_i / M$$

$$= \sum_{i=1}^{k} M_i \lambda_i [1 - E_2(N_i, \lambda_i, \mu)] / (M\mu) \qquad 3.2.11$$

From the above definition, an optimum assignment of M_i, λ_i and N_i to minimize B will in general differ from that required to maximize R. Instead of creating a larger buffer, more small buffers will be created. Although this may decrease the total blocking probability, it also decreases the total buffer utilization. In addition, minimizing the total blocking probability may favor small packets. In some situations the optimum assignment of N_i will give a blocking probability of 1 to large packets. Before a better criterion can be found, the blocking probability and buffer utilization factor will be used for evalution. Almost all buffer management schemes favor packets of small size over packets of large size. Figures 3.2.2 and 3.2.3 show the optimum assignment of N_1 and N_2 for both minimizing B and maximizing R respectively. Two size buffers are considered, one of size M_1 and one of size M_2=D, the maximum size of a packet. A uniform distribution of packet size is assumed, i.e. $\lambda_1=\lambda M_1/D$, and $\lambda_2=\lambda-\lambda_1$, where λ is the total input rate of packets. The number in the bracket of the figure is the corresponding number of N_1 and N_2 for the optimization assignment. Figure 3.2.4

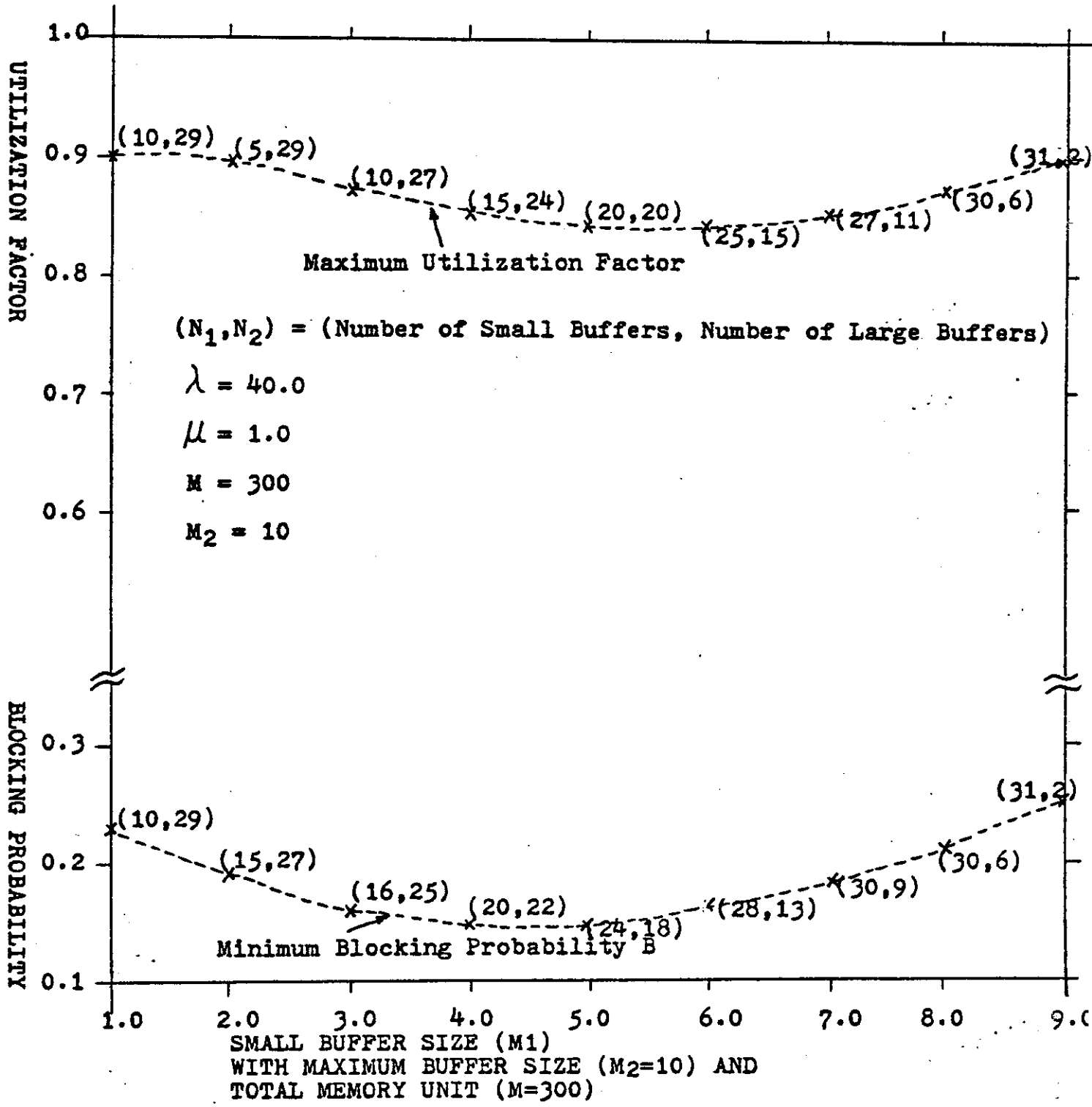


FIGURE 3.2.2 STATIC BUFFER ASSIGNMENT AT HEAVY LOAD

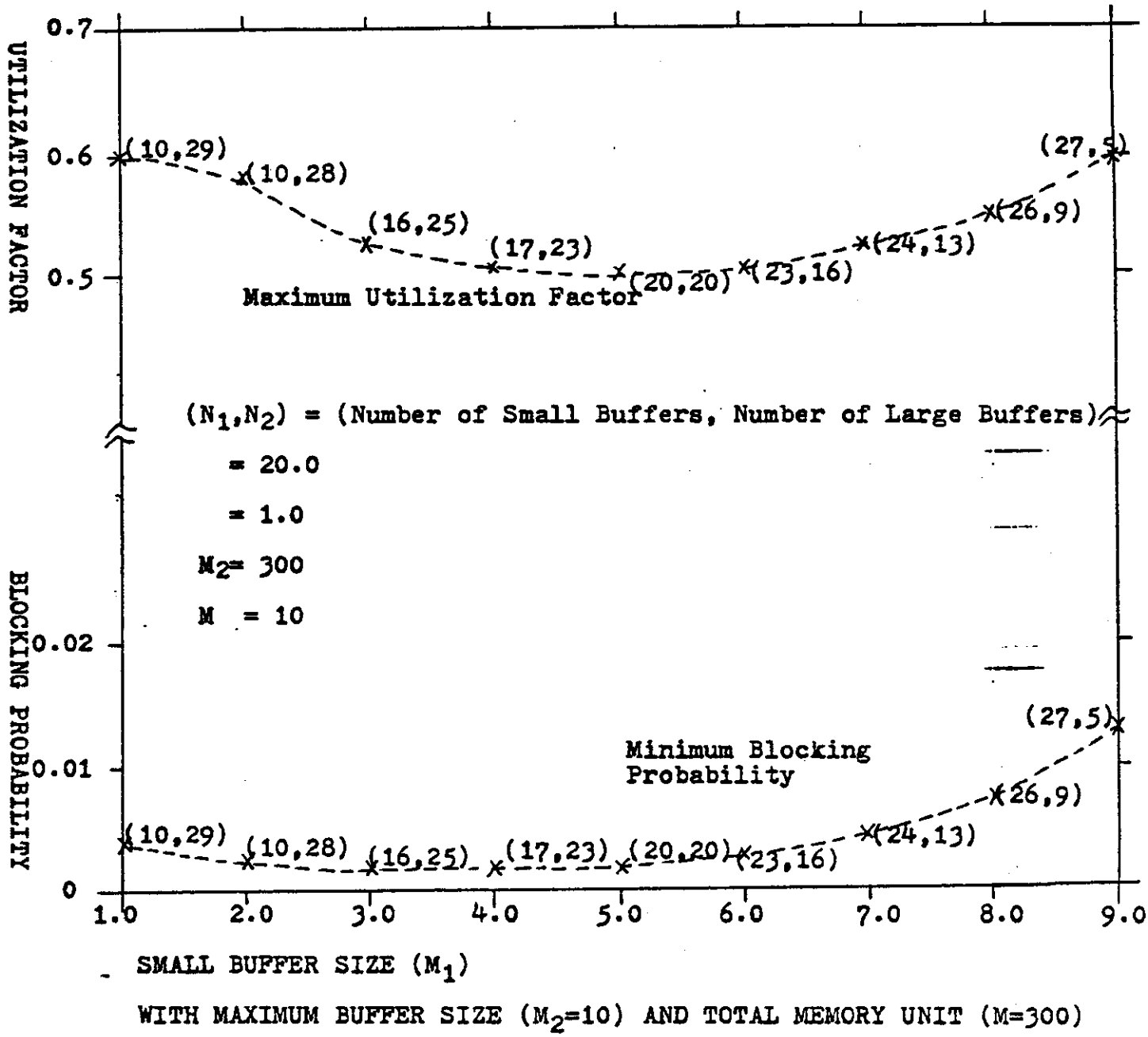


FIGURE 3.2.3 STATIC BUFFER ASSIGNMENT

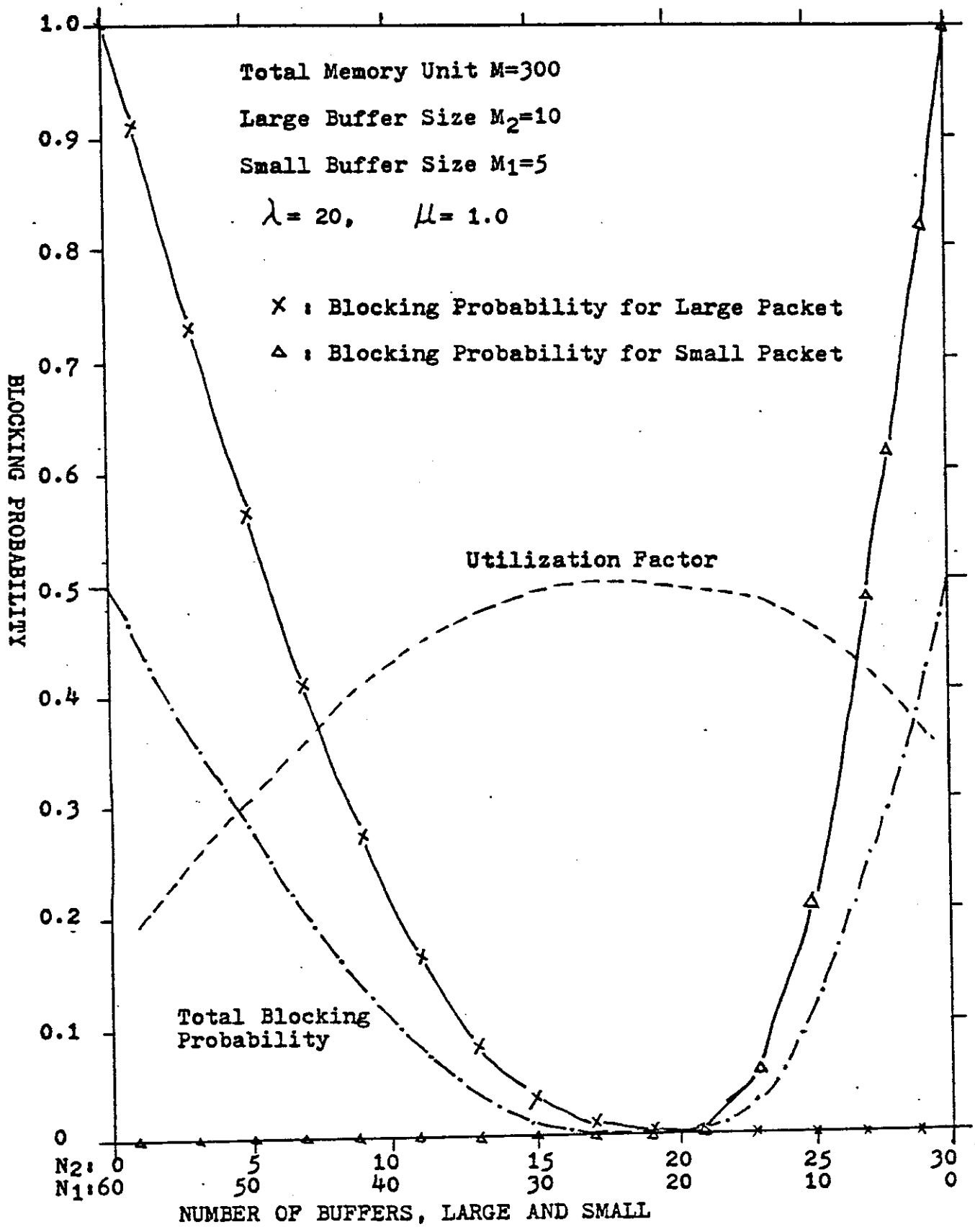


FIGURE 3.2.4 STATIC TWO SIZE BUFFER ASSIGNMENT

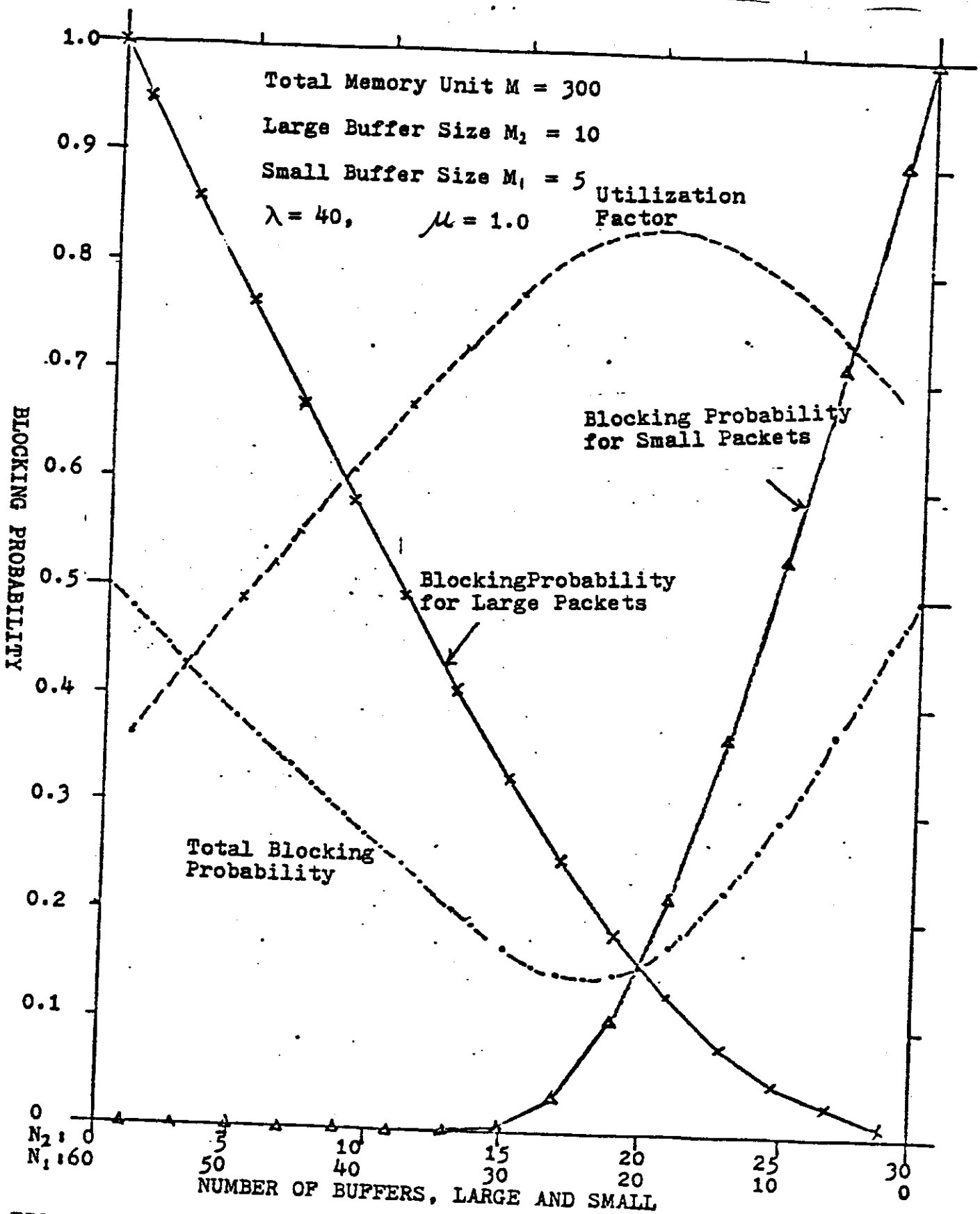


FIGURE 3.2.5 STATIC TWO SIZE BUFFER ASSIGNMENT AT HEAVY LOAD

and 3.2.5 show the blocking probabilities and utilization factor when we have M_1 equal to D/2. Comparing the three figures, we see that the choice of the size of M_1 is not very critical as long as an optimum combination of N_1 and N_2 can be found. But for fixed M_1, the choice of N_1 and N_2 is rather critical. For a lightly loaded system, the optimization assignment of N_1 and N_2 are the same for both minimizing B and maximizing R.

Dynamic Assignment

Now we consider the system that allows large buffers to be assigned to small packets if no small buffers are available. The definitions of M_j, N_j and λ_j used in static assignment are also used here. A small packet is blocked only if all the buffers are occupied. A large packet is blocked if no large buffer is available. Apparently, the system is heavily biased in favor of packets of small size. There are many reasons for using such system. For example, as we shall see later, in addition to being relatively simple, it gives better throughput, a lower total blocking probability, and a higher buffer utilization factor. As we will see later, in some situations this buffer management scheme is better than the static assignment scheme.

A two-dimensional Markov Chain is built for a system with two buffer sizes, large and small. The state space of the Markov Chain is $S=\{(i,j) \mid i,j \geq 0\}$, where i,j indicates the number of small and large buffers occupied, respectively. Notice that the state is not defined by the number of packets of relative size, because some small packets may occupy large buffers. No reassignment of buffers will be made if some small buffer is freed after a small packet has already been assigned a large buffer. Figure 3.2.6 shows the probability transition flow of the Markov chain. The horizontal

transition rates, i.e., transition between (i,j) and (i,j+1), are λ_2 and $(j+1)\mu$, except for the last row, where i=N, and the large buffer will be assigned to the small packets too.

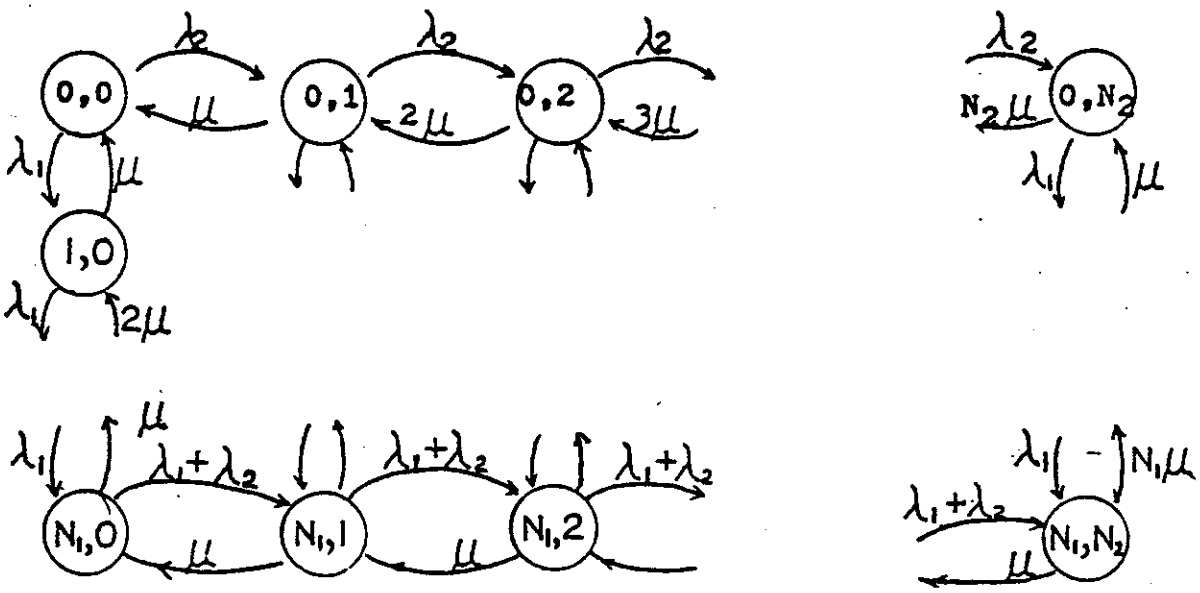


FIGURE 3.2.6 STATIC SPACE OF DYNAMIC ASSIGNMENT OF BUFFERS

The balance equations are

$$[\lambda_1+\lambda_2+(i+j)\mu]\, P_{i,j} = \lambda_1 P_{i-1,j} + \lambda_2 P_{i,j-1} + (i+1)\mu P_{i+1,j} + (j+1)\mu P_{i,j+1}$$

for $0<i<N_1$, $0<j<N_2$ 3.2.12

for i=0, j=0, the first and second term of the right hand side vanish, respectively.

$$[\lambda_1+\lambda_2+i\mu]\, P_{i,0} = \lambda_1 P_{i-1,0} + (i+1)\mu P_{i+1,0} + \mu P_{i,0}$$

for $0<i<N_1$

and

$$[\lambda_1+\lambda_2+j\mu]\, P_{0,j} = \lambda_2 P_{0,j-1} + \mu P_{1,j} + (j+1)\mu P_{0j+1}$$

for $0<j<N_2$

$$[\lambda_1 + \lambda_2]\, P_{0,0} = \mu P_{1,0} + \mu P_{0,1}$$

for $j = N_2$, the last term vanishes,

$$[\lambda_1+(i+N_2)\mu]\, P_{i,N_2} = \lambda_1 P_{i-1,N_2} + \lambda_2 P_{i,N_2-1} + (i+1)\mu P_{i+1,N_2}$$

Note that for $i = N_1$, the transition flow is different. There is no small buffer

available, so any new small packets will be assigned a large buffer until all larger buffers are also occupied. The transition rate from (N_1,j) to $(N_1,j+1)$ is $(\lambda_1+\lambda_2)$, instead of λ_2. And the balance equation becomes:

$$[\lambda_1+\lambda_2+(N_1+j)\mu]P_{N_1,j} = \lambda_1 P_{N_1-1,j} + (\lambda_1+\lambda_2)P_{N_1,j-1} + (j+1)\mu P_{N_1,j+1}$$

$$\text{for } 0<j<N_2 \qquad 3.2.13$$

And for $j=0$ and $j=N_2$, the balance equations become:

$$[\lambda_1+\lambda_2+N_1\mu]P_{N_1,0} = \lambda_1 P_{N_1-1,0} + \mu P_{N_1,1}$$

$$(N_1+N_2)\mu\, P_{N_1,N_2} = \lambda_1\, P_{N_1-1,N_2} + (\lambda_1+\lambda_2)\, P_{N_1,N_2-1}$$

The above set of equations can be solved by the constraint that $P_{i,j}$ are probabilities, i.e., $\sum_i \sum_j P_{ij} = 1$. A straight-forward Gauss-Seidel algorithm [Ral69] is used to solve the system, and some results are shown in figure 3.2.7 and 3.2.8. The blocking probability for small packets, B_1, is equal to the probability P_{N_1,N_2} and the blocking probability for large packets is equal to summation of P_{i,N_2} over all i.

The Gauss-Seidel algorithm needs a space of $(N_1+1)X(N_2+1)$, and its convergent rate depends also on N_1 and N_2. For N_1, N_2 less than 100, the algorithm will converge to an error of less than $10^{(-4)}$ in less than several minutes of CPU time on a PDP-10. But for large N_1, N_2 or of system with k different size buffer, there will be some difficulty in calculating the exact probability distribution. Two approximation formula are suggested below.

Let us model the system as seen from the perspective of the large buffers. During one period the stream of requests is Poisson input of rate $\lambda_1+\lambda_2$, while during the other period the request stream has an input rate of only λ_2. Figure 3.2.9 depicts its input rate of the queueing system.

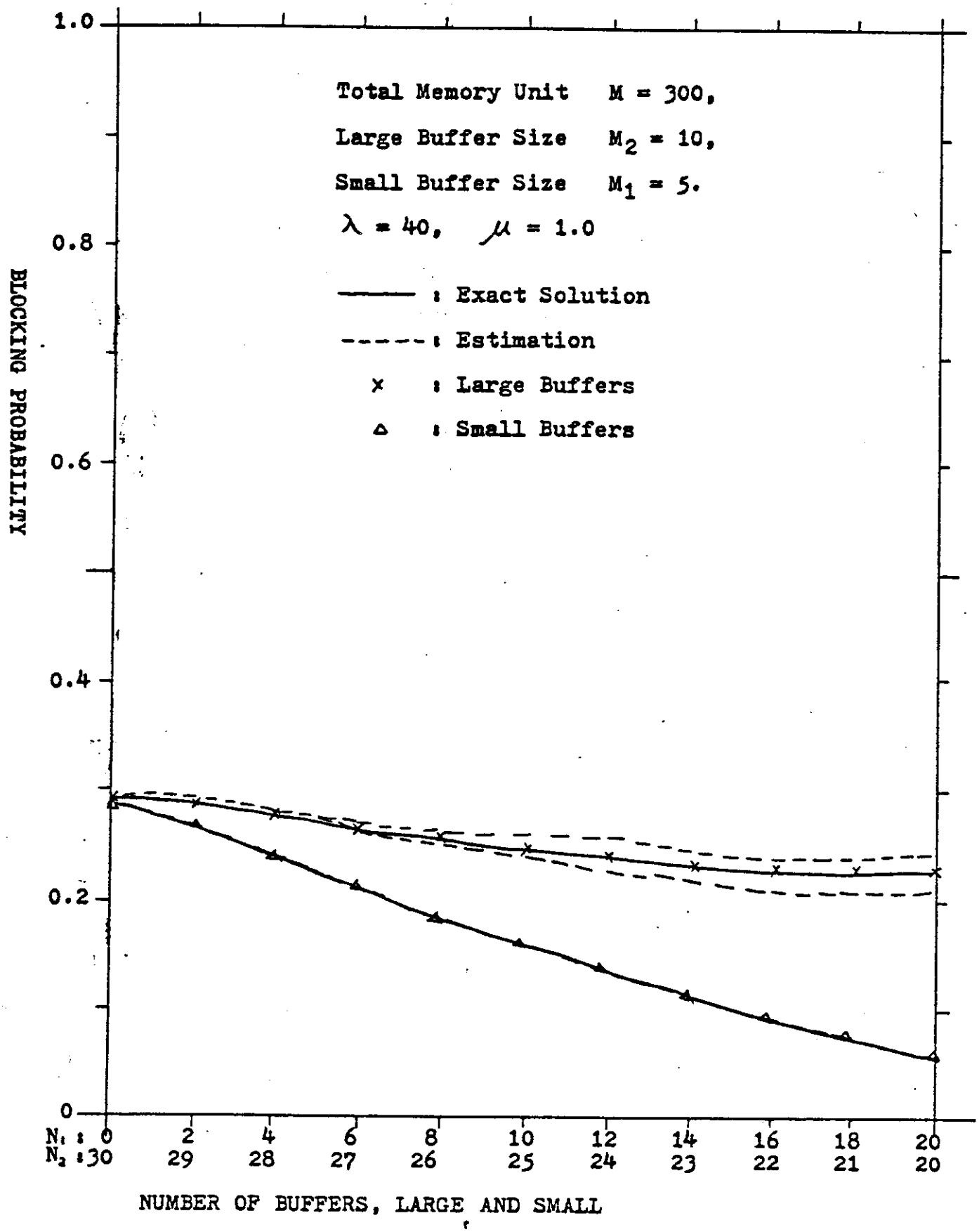


FIGURE 3.2.7 DYNAMIC ASSIGNMENT OF TWO SIZE BUFFER

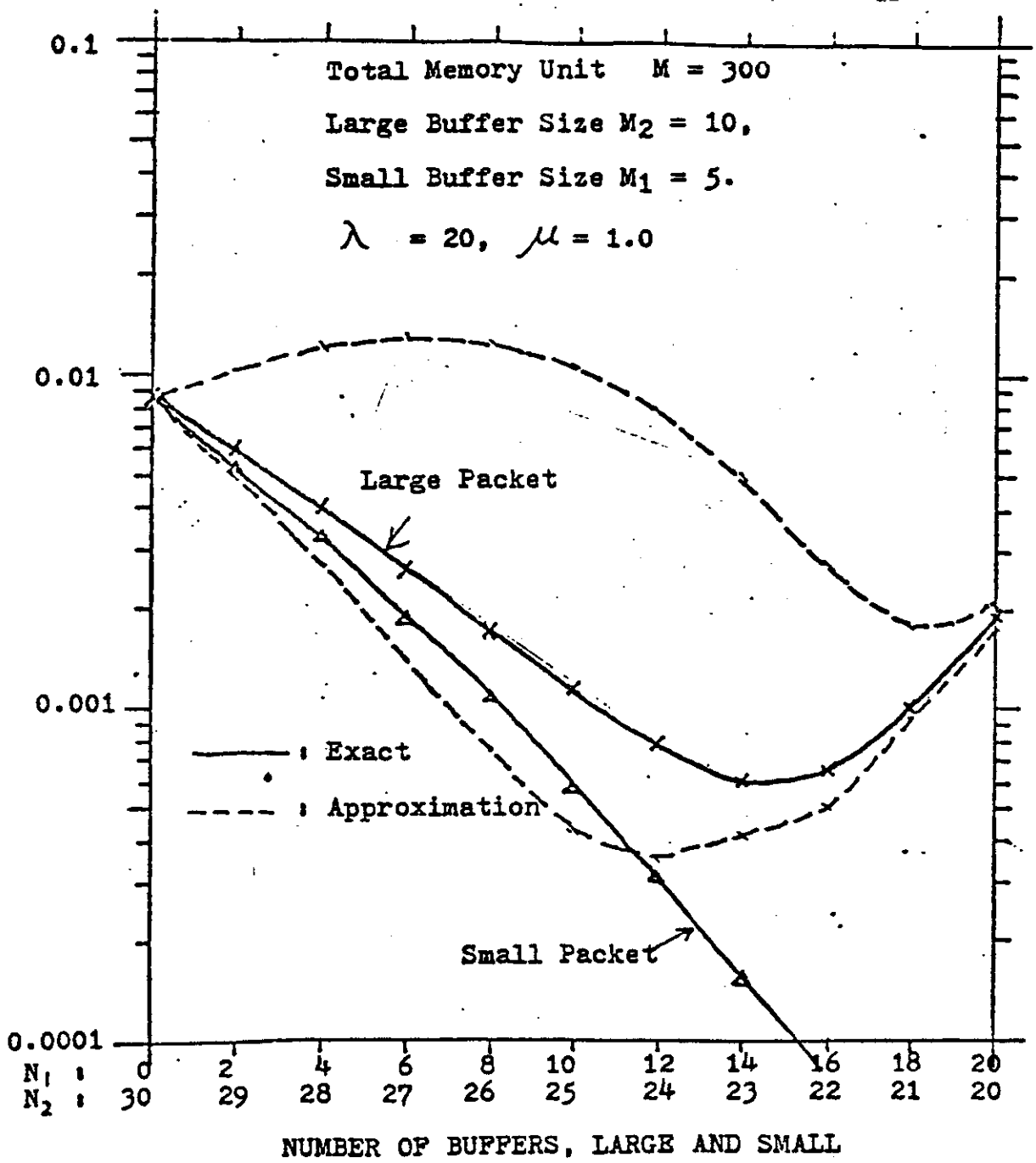


FIGURE 3.2.8 DYNAMIC ASSIGNMENT FOR TWO SIZE BUFFER AT LIGHT LOAD CONDITION

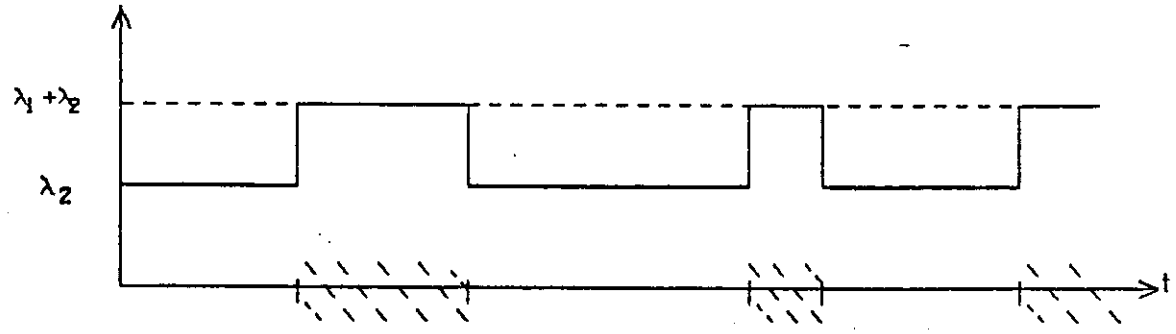


FIGURE 3.2.9 INPUT RATE OF THE QUEUE

During the period indicated by shaded area, the request rate for large buffers increases to $(\lambda_1+\lambda_2)$; otherwise the rate remains λ_2. The shaded area depicts the busy period for the $M/M/N_1/N_1$ queueing system of small buffers. In general this period will not be exponentially distributed, which implies that the system has quite complicated input states. We know, however, that the ratio of the shaded period to the whole period is $x=E_2(N_1,\lambda_1,\mu)$, the busy probability for the $M/M/N_1/N_1$ queueing system of small buffers; If we keep this ratio x constant, but shrink or expand the duration of the shaded period and the non-shaded period, two approximation formulas are derived:

(P1) Keep the ratio x constant and let the duration of the average shaded period go to zero. The system becomes an $M/M/N_2/N_2$ queueing system with the same service rate but an input rate of $(\lambda_2+\lambda_1\cdot x)$. Thus the blocking probability for large packets becomes $E_2(N_2,\lambda_2+\lambda_1\cdot x,\mu)$.

(P2) Keep the ratio x constant and let the duration of the shaded period go to infinity. The system will have a negligible transience from one $M/M/N_2/N_2$ queueing system of input rate λ_2 to another $M/M/N_2/N_2$ queueing system of input rate $(\lambda_1+\lambda_2)$ and back and forth. The blocking probability for large buffers is the average of these two : $xE_2(N_2,\lambda_1+\lambda_2,\mu) + (1-x)\,E_2(N_2,\lambda_2,\mu)$.

Figures 3.2.7 and 3.2.8 show the blocking probability for both small packets and large packets and the two approximation formulas. When the system is lightly loaded, the values of the above two formulas will be quite far away from each other, as shown in Figure 3.2.8, and the exact value will lie somewhere between the extremes. But for a moderately loaded system, i.e., λ=40, the values of the approximation formula are quite close, and both are good approximations of the exact solution, as Figure 3.2.7 shows.

Comparing Figure 3.2.8 with Figures 3.2.2, 3.2.3 and 3.2.4, for lightly loaded systems, we find that the dynamic assignment system becomes the superior scheme. In the moderately loaded system, however, Figures 3.2.7 and 3.2.5 show the value of the dynamic assignment is questionable. For a lightly loaded system, λ = 20, the optimum assignment of N_1, N_2 for static assignment will get a blocking probability of 0.00187 for both large and small packets. For dynamic assignment, N_1=14 and N_2=23 will give a blocking probability about three times better for packets of both sizes. For a moderately loaded system, λ = 40, the dynamic assignment is strongly biased in favor of small packets. Large packets will always suffer a high blocking probability. For dynamic assignment, B_2, blocking probability for large packets, can never be lower than 0.21. For the static assignment, the blocking probability can be B_1=B_2=0.16 for both small and large packets. Thus if the system needs a relatively balance blocking probability or has an upper limit blocking probability for all packets, then dynamic assignment may not be proper under some loading condition.

3.2.3. Dynamic Memory Allocation

There are many similarities between buffer management in communication

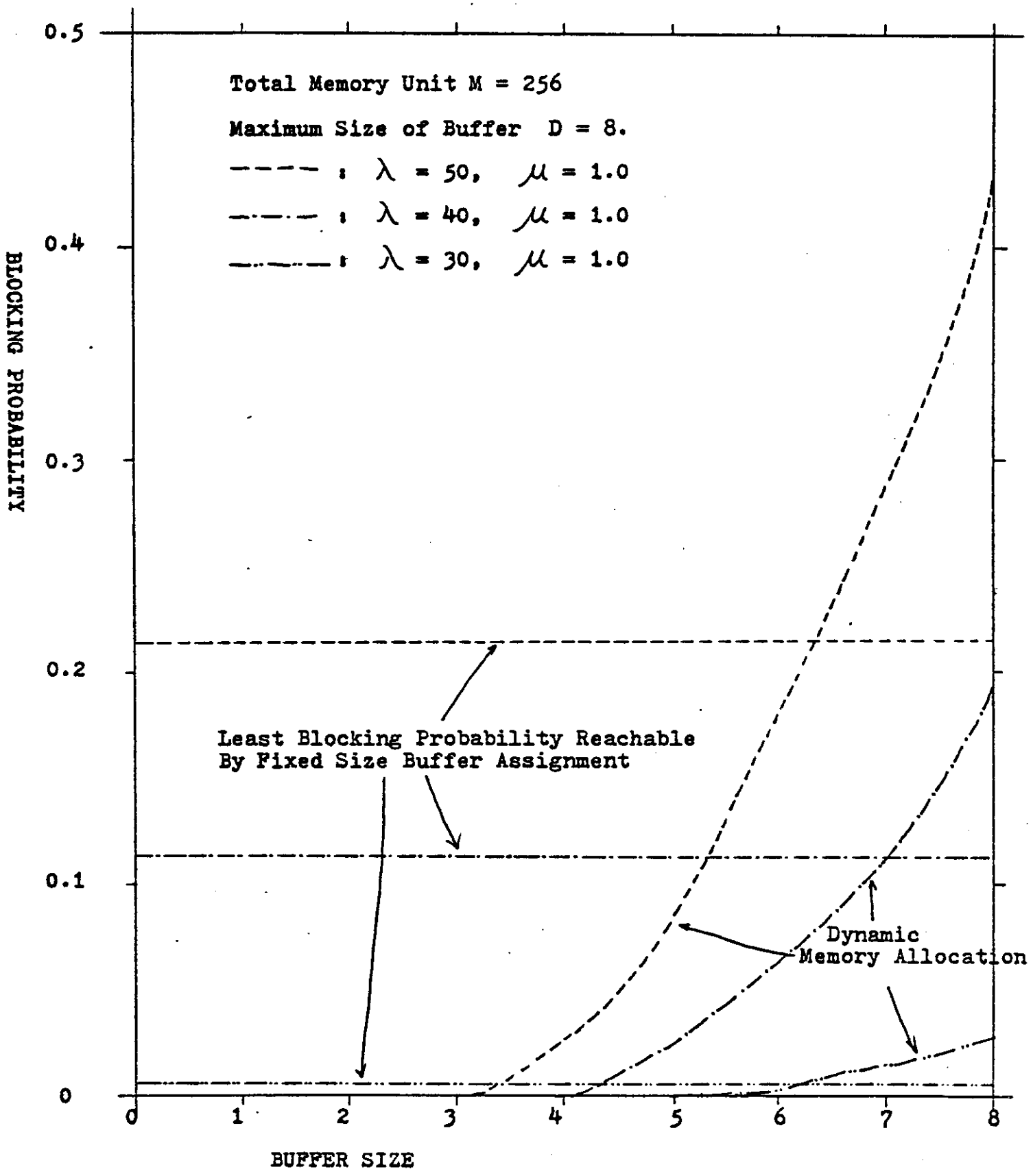
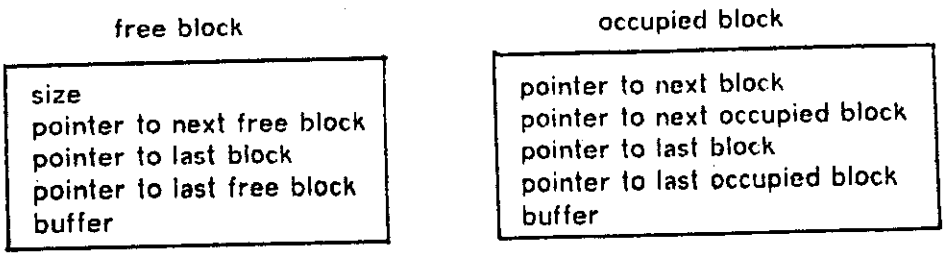


FIGURE 3.2.10 COMPARISON OF FIXED SIZE BUFFER ASSIGNMENT AND DYNAMIC MEMORY ALLOCATION

systems and memory management in multi-programming computer system. All buffer management schemes discussed here are very simple, because of the strict real time requirement in the communication environment. Now we will consider a more complicated memory management commonly used in the large computer system: dynamic memory management. There are many well-known algorithms, such as first-fit, best-fit, buddy, etc.[Knu68]. Each has its own characteristics and fits best into a specific environment. For the communication system in which simplicity and reliability are the most important requirements, a first-fit algorithm is tried and simulated for the buffer management of the communication system.

The first-fit algorithm used here is very similar to that mentioned in Volumne 1 of Knuth[Knu68]. The whole memory is divided into blocks dynamically. A block is a set of contiguous memory locations which are either free or belong to the same packet. There are two types of blocks, free or occupied, and every block is linked to another to form a set of ring buffers. There are four fields for each block: block size or starting address of next block, starting address of the next same type block, starting address of last block, and starting address of the last same type block.



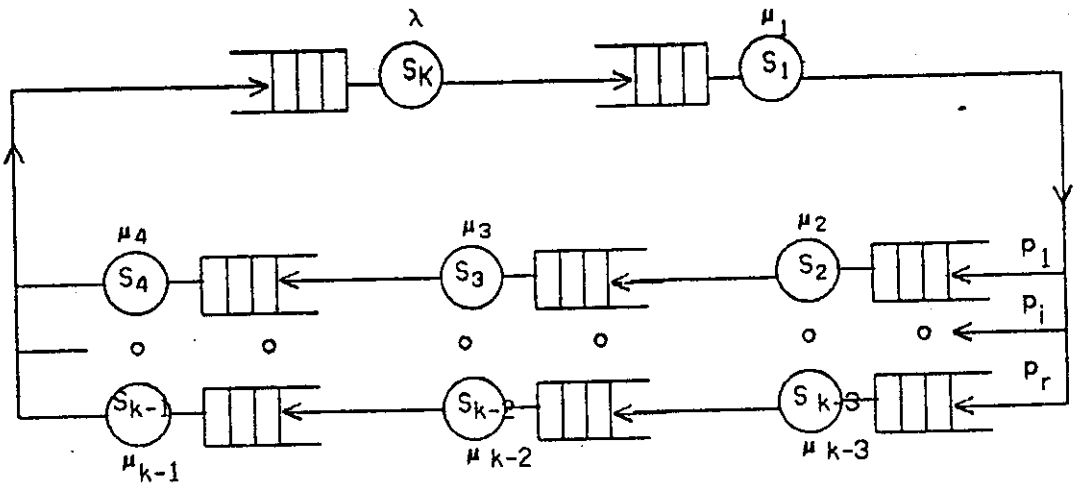
All the buffers, free or occupied, are linked together. Another pointer, AROV, points to the currently free block, and from this place the next incoming packet will begin to search for the first large enough free block on the free buffer ring. Because

of the ring structure, two flags, FEMP and BEMP, are needed to point out whether there is any free block or occupied block in the system. FEMP=1 if there is no free block, and BEMP=1 if there is no occupied or bounded block.

The detailed algoirthm can be found in [Knu68]. Only the blocking probabilities are compared here to other buffer managements techniques. The simulation results, Figure 3.2.10, shows the blocking probability for packets of different sizes. This scheme also favors smaller size packets over larger ones. This complicated memory management scheme offers a lower total blocking probability than all other schemes we have discussed, but it is also the most complex one.

3.3. Network Model

In the last section some buffer management techniques were tested. The life time of buffers is assumed to have an Exponential distribution with mean $\mu_i=i\mu$. This assumption will be justified, and the conditions under which the assumption holds will be also discussed in this section. First we will explain the model as follows:



A closed queueing network is used to model the buffer behavior of the current integrated switch. The number of customers in the queueing network is the total number of buffers, occupied or free, in the integrated switch. Customers in service center S_K are the free buffers of the switch; all other customers are buffered data packets. Customers in service center S_1 are packets which have been assigned buffers and are awaiting the processing in the current integrated switch. Those buffered packets are then transmitted, with probability p_1, through transmission line S_2, to the next integrated switch, S_3, with probability p_r, through S_{K-3}, S_{K-2}, etc. After the customers undergo the necessary processing in the next switch, S_3, the

corresponding positive acknowledgements are sent back, through S_4, to the current switch. When the current switch receives an acknowledgement, the buffer of the packet begin acknowledged is released and a new free buffer is created in S_K. When the number of customers in S_K is also zero, i.e., there is no free buffer in the current switch, the output rate of S_K is zero. This implies that all new incoming packets will be blocked because there are no free buffers, so it looks as if there are no new incoming packets. The service discipline for all service centers is FIFO (First In First Out). There may be one or more servers in a service center, and each service center may have different service distribution.

First the exponential servers are assumed, then a network mixing two kinds of customers is modeled and simplified. Later in the section, the service centers, S_2 and S_4 are modeled as delay lines rather than ordinary queues.

3.3.1. Exponential Queueing Network

Suppose there is a total of M customers in the closed queueing network and K service centers, S_1, S_2, ·, ·, S_K with c_1, c_2, ·, ·, c_K servers, respectively. One customer can not be serviced by more than one server, and has no preference for any server, since all servers in the same service center are identical. The service time at service center S_i is an exponentially distributed random variable with mean $1/\mu_i$, i=1, 2, ·, K. All the customers are identical too, and the state of the system is the number of customers in the service centers. The steady state of this kind of exponential queueing network was solved by Gordon and Newell [Gor67]. The probability that the system is in a specific state has the product form. Let n_i be the number of customers at service center S_i. Let the state of the system be $(n_1, n_2, \cdot, \cdot, n_K)$. Then the equilibrium probability that the system is at state $(n_1, n_2, \cdot, \cdot, n_K)$ is

$$P[n_1,n_2,\cdot,\cdot,n_K] = \frac{1}{G(M)} \prod_{i=1}^{K} (X_i)^{n_i} / A_i(n_i) \qquad 3.3.1$$

where $\sum_{i=1}^{K} n_i = M$ and

$$\mu_j X_j = \sum_{i=1}^{K} \mu_i X_i p_{i,j} \qquad j = 1,2,3,\cdot,\cdot,K.$$

where $p_{i,j}$ is the probability that a customer will proceed to the jth service center after completing a service request at the ith service center, and $1/\mu_i$ is the average service request at service center S_i. G(M) is the normalization constant so that the summation of $P[n_1,n_2,\cdot,\cdot,n_K]$ over all feasible states is one, where the feasible states are defined such that $\sum_{i=1}^{K} n_i = M$ and $n_i \geq 0$ for all $i=1,2,\cdot,\cdot,K$. That is

$$G(M) = \sum_{n \in S(M,K)} \prod_{i=1}^{K} [(X_i)^{n_i} / A_i(n_i)] \qquad 3.3.2$$

where $S(M,K) = \{(n_1,n_2,\cdot,\cdot,n_K) \mid \sum_{i=1}^{K} n_i = M \text{ and } n_i \geq 0 \text{ for all } i\}$, and $A_i(n_i)$ is defined recursively as follows:

$$A_i(0)=1$$

$$A_i(j) = j\, A_i(j-1) \qquad \text{if } j<c_i$$

$$A_i(j) = c_i\, A_i(j-1) \qquad \text{If } j \geq c_i \qquad 3.3.3$$

where c_i is the number of servers at service center S_i.

An algorithm was derived by Buzen [Buz73] to solve the above problem. A temporal variable, g(m,k), is defined as:

$$g(m,k) = \sum_{n \in S(m,k)} \prod_{i=1}^{k} (X_i)^{n_i} / A_i(n_i) \qquad 3.3.4$$

Note that G(M) = g(M,K), and in fact g(m,K) = G(m) for m=0,1,2,·,· M. From the above equation:

$$g(m,1) = X_1^{\,m} / A_1(m) \qquad \text{for } m=0,1,2,\cdot,\cdot,M \qquad 3.3.5$$

and $g(0,k) = 1$ for $k=0,1,2,\cdot,\cdot,K$

with the recursive formula of g(m,k):

$$g(m,k) = \sum_{j=1}^{m} [(X_k)^j / A_k(j)] \cdot g(m-j,k-1) \qquad 3.3.6$$

g(m,k) can be calculated. Only 2(M+1) memory space are needed to store g(m,k-1) and to calculate g(m,k) for m=0,1,2,3,..., M.

The marginal probability, $P_K(j)$, is calculated

$P_K(j)$ = j customers at service station S_K.

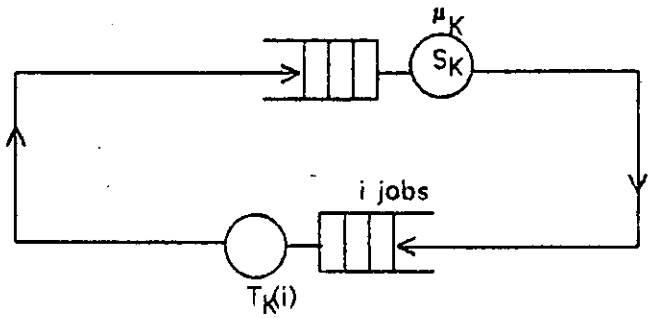
$$= [(X_K)^j / A_K(j)] \cdot \frac{g(M-j,K-1)}{G(M)}$$

$$= [(X_K)^j / A_K(j)] \cdot \frac{g(M-j,K-1)}{g(M,K)} \qquad 3.3.7$$

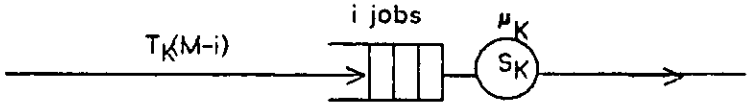
Define throughput $T_K(m)$ as the rate of which customers pass through service center S_K when there are m customers in the queueing network and the service time at service center S_K is reduced to zero. Then $T_K(m)$ becomes:

$$T_K(m) = X_K \cdot g(m-1,K-1) / g(m,K-1) \qquad 3.3.8$$

By Theorem 1 of [Cha75], the effect on the queueing network with regard to service center S_K is equivalent to the following load-dependent queueing network.



$T_K(i)$ is the load dependent input rate of a single queue.



where $T_K(0)=0$. The probability that a service center S_K has i customers is:

$$P_K(i) = P_K(0) \cdot \prod_{j=0}^{i-1} [T_K(M-j) / \mu_K(j)] \qquad 3.3.9$$

where

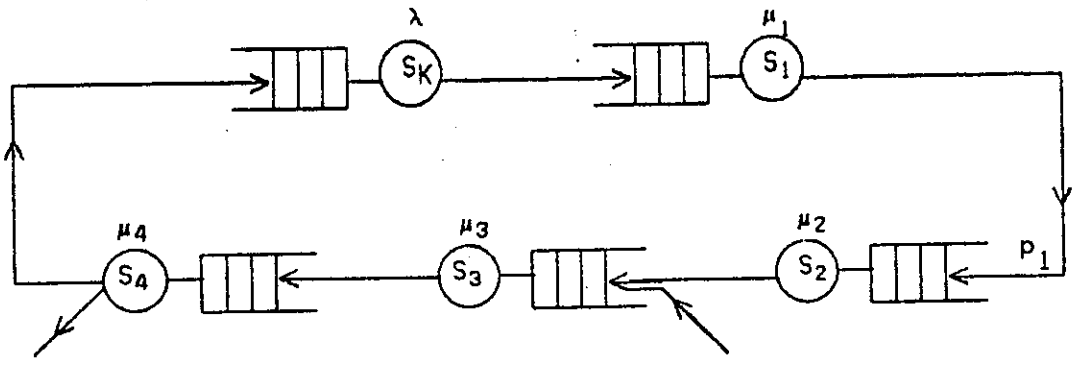
$$P_K(0) = [1 + \sum_{i=1}^{M} \prod_{j=0}^{i-1} \{T_K(M-j) / \mu_K(j)\}]^{-1}$$

which are exact in the same form of equation (3.2.2) and (3.2.3).

From the above derivation, we know that $T_K(j)$ works as an interface between service center S_K and the rest of the network. If service center S_K is the free buffer queue, then $[1/T_K(1)]$ is the average life time of an occupied buffer, and the assumption of last section can be simplified as

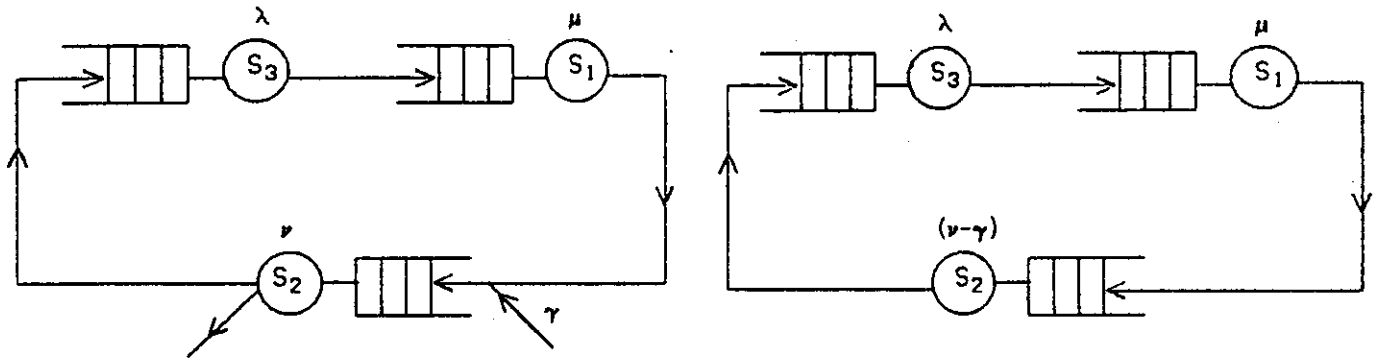
$$T_K(i) = i \cdot T_K(1) \qquad 3.3.10$$

Now we consider the more complicated network in which the integrated switch S_3 has other work to do besides processing the packets coming from integrated switch S_1. Here there are streams of jobs flowing into switch S_3, requesting service, and leaving the queueing network. The queueing network becomes:



There are two kinds of customers. One kind includes the buffers of the current integrated switch and the corresponding packets; the other kind includes those

requesting service to switch S_3 other than the packets sent from switch S_1. S_2 and S_4 are assumed to be the trunk lines between integrated switch S_1 and S_3. S_K is the free buffer queue of the current integrated switch S_1. Let us represent the network as the left network of the following figure, and prove that the two networks are identical as far as the first kind of customers is concerned.



Let i_1, i_2, i_3 be the number of customers of first kind in service centers S_1, S_2, S_3, respectively. Let j_2 be the number of customers of second kind in service center S_2. The service requests for both kinds of customers are the same exponentially distributed random variables. The service discipline for both kinds of customers is the same FIFO. Then, by the technique of independent balance equation developed by Baskett et al. [Bas73], the equilibrium state probabilities are of product form:

$$P[i_1, i_2, i_3, j_2] = D\, h_1(i_1)\, h_2(i_2, j_2)\, h_3(i_3) \qquad 3.3.11$$

where $(i_1+i_2+i_3)=M$, the total number of customers of the first kind in the queueing network. And

$$\begin{aligned} h_1(i) &= (1/\mu)^i \\ h_2(i,j) &= (i+j)! \cdot \frac{1}{i!} (1/\nu)^i \cdot \frac{1}{j!} (\gamma/\nu)^j \\ h_3(i) &= (1/\lambda)^i \end{aligned} \qquad 3.3.12$$

where D is the normalization factor such that the summation of $P[i_1, i_2, i_3, j_2]$ over all feasible states is one.

$$D = [\sum_{i_1+i_2+i_3=M} \sum_{j_2=0}^{\infty} P[i_1, i_2, i_3, j_2]]^{-1} \qquad 3.3.13$$

The marginal distribution of $h_2(i_2)$ is

$$h_2(i) = \sum_{j=0}^{\infty} h_2(i_2, j_2)$$

$$= \frac{1}{i!} (1/\nu)^i \cdot \sum_{j=0}^{\infty} \frac{(i+j)!}{j!} (\gamma/\nu)^j \qquad 3.3.14$$

By the identity formula[Fel66],

$$\sum_{j=0}^{\infty} \frac{(i+j)!}{j!\,i!} y^j = \sum_{j=0}^{\infty} \binom{i+j}{i} y^j = (1-y)^{i+1} \qquad 3.3.15$$

so

$$h_2(i) = (1-\gamma/\nu)^{-i-1} \cdot (1/\nu)^i$$

$$= (1/\theta)^i \cdot (1 - \gamma/\nu)^{-1}$$

where $\theta = \nu - \gamma$, so

$$P[i_1, i_2, i_3] = D' (1/\mu)^{i_1} (1/\theta)^{i_2} (1/\lambda)^{i_3} \qquad 3.3.16$$

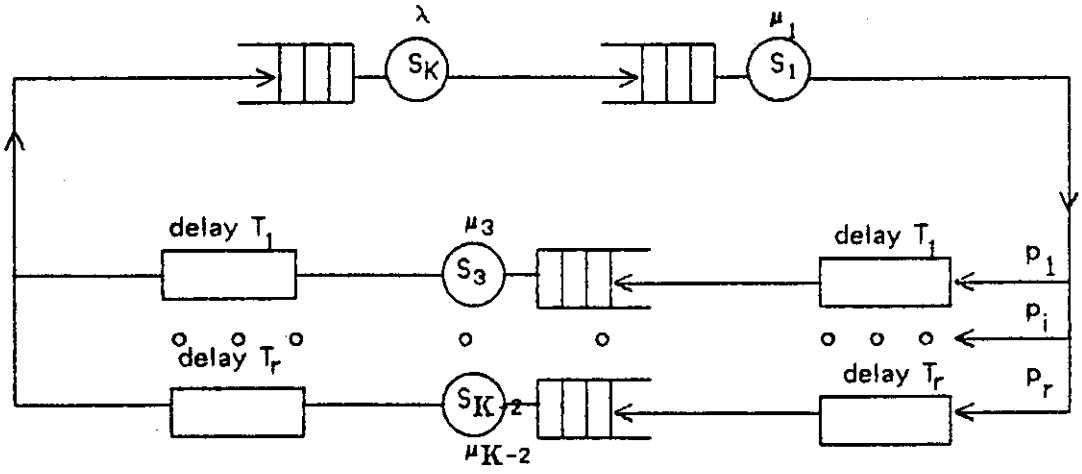
where $D' = D \cdot (1-\gamma/\nu)^{-1}$. The above formula just presents the system state probability of the queueing network at the right side.

Thus the external streams of requests to an exponential server of the closed queueing network will make the service rate decrease by the same amount. Then the pure closed queueing network can be solved.

3.3.2. Queueing Network with Time Lag

SENET is a special implementation for an integrated switch. The most important character is the frame format. According to the implementation at Carnegie-Mellon

University [Bar76], the switch will have frame buffers. There is a finite delay for the packing of a frame, transmitting it, and unpacking it. The delay is at least two frame periods, no matter how much processing power is available. Also there is the time required for transmitting a packet from one switch to another. Transmission line, S_2, S_4, ·, · of the queueing network are actually more like a delay line. The queueing network is redrawn as follows:



Service centers S_1,S_3 , ·, ·, S_{K-2} are defined as integrated switches. After a service request is completed, a customer, which is a buffered packet, is put into the output frame of the switch. After some delay, the packet will appear in the input frame of the next switch. The delay is defined as the period extending from the moment the packet is copied into the output frame to the moment the packet is copied from the input frame of the next switch. If the largest share of a buffer's life time is spent waiting for service at integrated switches, then the input rate of service center S_K will be limited by the service rate of the integrated switch. On the other hand, if the delay is much longer than the waiting time and service time in the integrated switches, then $T_K(i)$ will be proportional to i.

The delay line can be modeled as an M/G/∞ queue. The customer encounters no waiting time; service time is the delay. Closed finite queueing networks with this kind of delay was studied by Posner and Bernholtz[Pos68]. The derivation is very involved, but the result is quite simple and powerful. The result has the same product form as the exponential queueing network, but the delay may not be exponentially distributed. Only the average delay enters the formula.

$$P[n_1,n_2,\cdot,\cdot,n_K] = D\, h_1(n_1)\, h_2(n_2) \cdots h_K(n_k) \qquad 3.3.17$$

If jth station is a delay line of average delay T_j, then

$$h_j(n_j) = (T_j)^{n_j} / n_j! \qquad 3.3.18$$

Define p_i as the probability that a packet will go to the corresponding i switch when it finishes the processing of the current integrated switch. The equation can be more simplified.

$$h_d(n_d) = T^{n_d} / n_d! \qquad 3.3.19$$

where n_d is the total number of customers in delay lines rather than in the integrated switch, and

$$T = \sum_{i=1}^{r} p_i T_i$$

is the average total delay. The state probability of n_i customers in ith integrated switch becomes

$$P[n_i, \text{ for all i such that } S_i \in \{\text{switch}\}]$$
$$= D\left[\prod_{i \in S} h_i(n_i)\right] h_d(n_d) \qquad 3.3.20$$

where $\sum_{i \in S} n_i + n_d = M$, total number of customers in the network. S is the set of service centers of the queueing network which are referred to the integrated switch of the real communication system.

3.3.3. Numerical Results

Figure 3.3.1 shows the throughput with respect to i. Here the delay line model of the last section is used. The delay time is set to be 15 milliseconds. The curves show the processing capability for the switching processors. When the switching processor can process four packets during each 10 millisecond frame period, the throughput is almost linear. If the switching processor can only process one packet for each frame period, the throughput reaches a maximum value very rapidly. The throughput is now limited by the switching processor and most of the buffers are queued in the system and waiting for processing rather than being transmitted in the communication links.

T(i) increases almost linearly with respect to i when i is small; it rapidly approaches to a limit value when i is large. T(i) can be approximated by the following formula.

$$T(i) = \text{minimum of } (ib, \tau) \qquad 3.3.21$$

where 1/b is the summation of average service time for a packet through the network.

There is a physical interpretation for this. When i is small, the packets are widely scattered in the queueing network. The chance that a packet has to wait for another packet is very small, so the throughput of the network increases linearly with respect to i. When i becomes large, the throughput is bounded by the processing capability of the network, so the increase in the number of packets will not increase the throughput. We will divide the curves into two regions: the customer-bounded region, $i \leq \tau/b$, and the processing-bounded region, $i > \tau/b$, respectively.

The model we discussed in the last section is assumed in the customer bounded region. In that region, the switch has enough processing power to process all the

incoming packets. The blocking of packets is due to the insufficient buffer spaces. If the integrated switch is in the processing-bounded region, the number of buffers is not very critical. In an ordinary packet switching network, if the system is in the processing-bounded region, the network is congested. For integrated switch, this condition is called the over-loaded state, i.e., $\lambda > (c - ic_v)$. In the next section we will discuss the use of a secondary storage in the processing-bounded region.

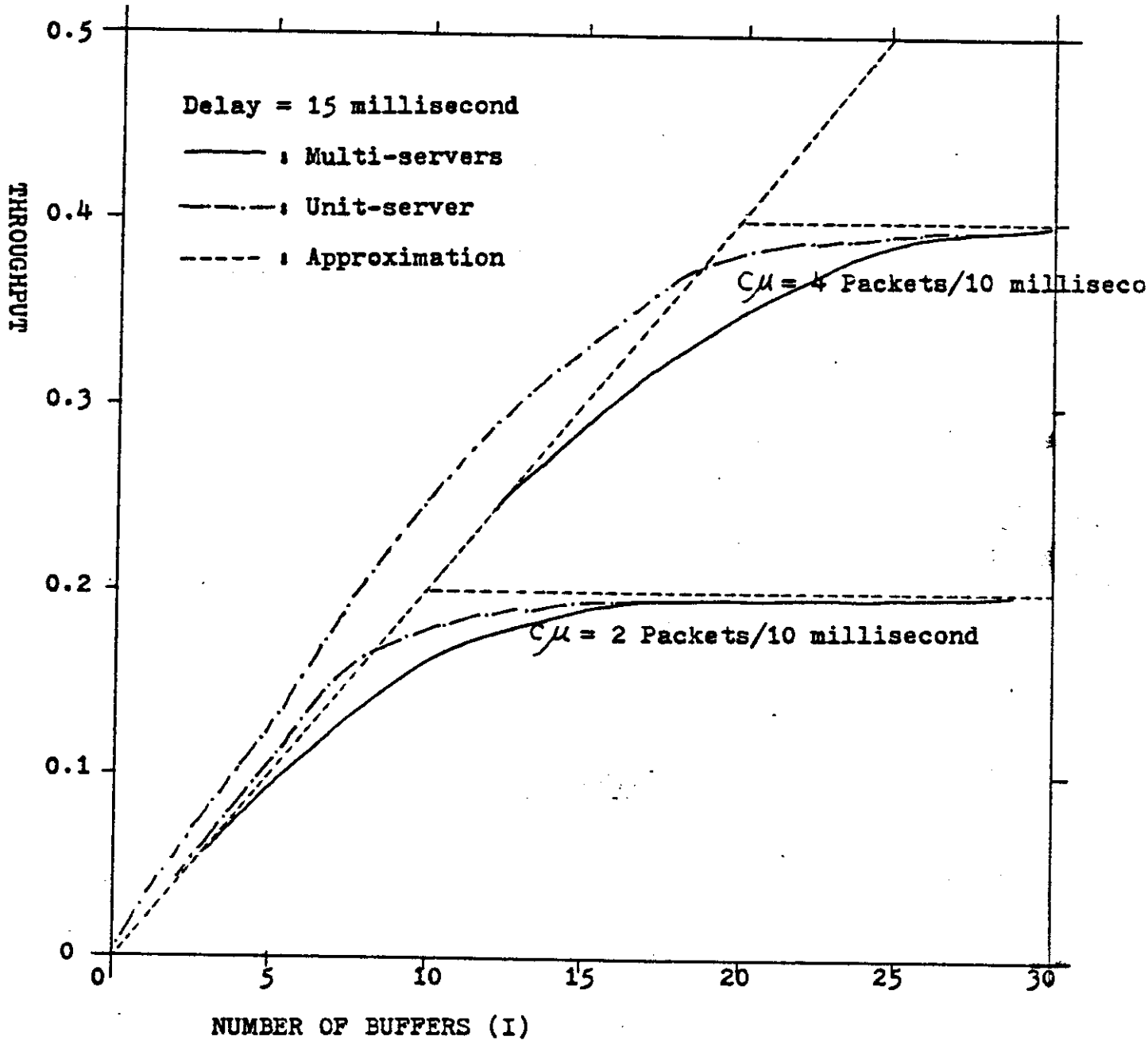


FIGURE 3.3.1 THROUGHPUT OF NETWORK QUEUEING MODEL

3.4. Secondary Storage Model

In general, a communication processor will have a disk or a secondary storage device if the processor has to manage a file system as well as perform its switching function. For a pure switching processor, however, the speed of a disk is too slow to meet the needs of a real time environment, so a disk is seldom used. For an ordinary packet-switching network, like ARPANET, a disk is not necessary. Because the expected waiting length of packets at the switch is much less than the buffer space of the switch, congestion will not occur frequently. Although the function of the integrated switch is similar to that of an ordinary packet-switching network, we feel that for the specific environment, a secondary storage device might be helpful. For the integrated switch, the waiting time has a very large variance, and the expected frequency of congestion is rather high. A congestion of an integrated switch occurs when it enters the overloaded states, i.e., when the input rate is higher than the processing capacity for the Class II packets. This will happen whenever i, the number of Class I customers, is above a certain level such that $(c-i*c_v)\mu_2<\lambda_2$. Once congestion occurs, not only will packets passing the congested switch be slowed down, but all the adjacent switches will be affected. All the packets destined for the congested switch cannot be sent out at regular rate, and the adjacent switches will have less buffer space and processing power to process other packets. When the cause of the congestion disappears, some time is required to release the congestion of the network. because the expected frequency of the integrated switch being in the overloaded states is rather high, a secondary storage device is suggested for the integrated switch. Such a device will prevent the adjacent switches from being affected by

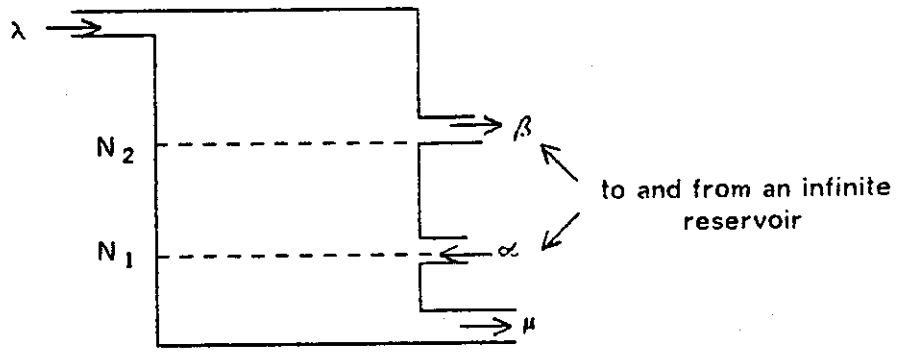
congestion and speed up the recovery process after congestion. When the integrated switch is in the overloaded state, it can pump the excess packets into the secondary storage device rather than block them. Although the packets will still suffer extra delay, time to transfer to the disk and back, an overloaded switch will not affect its neighbors by blocking the packets transmitted to it.

Whether to block a packet and let it be transmitted again or to accept it and put it into disk is unclear. For both cases, the packet will suffer extra delay. A four-frame period is usually required to make sure a packet is blocked and to transmit it again. The transfer time to and from a disk is of the same order of magnitude, so once the situation happens, it is going to suffer about the same amount of delay. There are merits to blocking a packet rather than putting it on the disk, however. This is especially true when the dynamic routing algorithm is used to determine the fastest route for a packet. It is hard to determine the response time if the next switch gives a quick positive acknowledgement but puts the packet in disk for a long period. On the other hand, the use of a disk will relieve the congestion once it occurs.

3.4.1. Modeling

A model similar to a water container is used to model the secondary storage device. The buffer space is modeled as the water container with input rate λ and output (service) rate μ. If the container is full, no more water can come in; if the container is empty, no water will flow out. Unlike an ordinary container with one input pipe and one output pipe, this container is connected to an infinite reservoir through two extra pipes at water levels N_1 and N_2. When the water level is below N_1, another water stream of rate α will flow into the container. When the water level is above N_2,

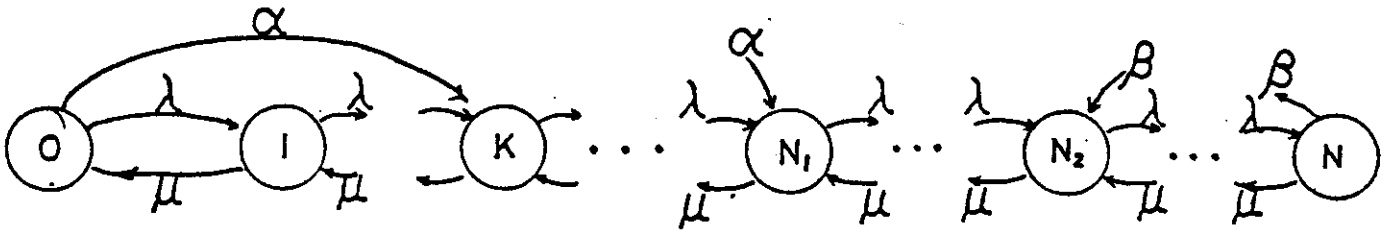
a water stream of rate β will be pumped out of the container. The purpose of these two extra pipes is to decrease the probability of the container being full and to increase the usage and the throughput of the container.



Returning to the integrated switch, we can let the water container stand for the buffer space and let the infinite reservoir stand for the disk, the secondary storage. β and α will represent the transfer rates to and from the disk. The secondary storage is used only to store the excessive data packets, and a full parallel processing of the switching processor and the disk controller is assumed. Whenever the number of packets exceeds the limit N_2, the disk will be enabled to transfer a block of data onto the disk. When the number of packets in main memory is below N_1, the reverse process occurs, i.e., a block of data is brought back into the main memory. A block of data moving to and from the disk may contain one or more data packets. This model reflects the nature of a communication processor. Unlike most I/O queueing models of large multi-programming computer, the data (packets) are not designated to any specific users. Like the water in our analogy, which makes no distinction between users, this model does not require that the user's space and the packets be monitored.

Every process is assumed to be memoryless, i.e., Poisson input process,

exponentially distributed service time and block transfer rates. A Markov chain model is built as follows:



The state of the system is the number of packets or the number of occupied buffers in the system, where λ is input rate of packets, $1/\mu$, the average service time, $1/\alpha$, the average time to read a block of data from the disk, $1/\beta$, the average time to write a block of data on the disk, K, the number of data packets in a block, and N, the number of buffers in the primary memory of the switch.

In the above model, a block transfer from the disk will be requested as long as the number of occupied buffers is less than N_1. The transfer request will be granted only if there are still less than N_1 occupied buffers at the time the block is completed. Similarly, a block transfer to disk request will be granted only if there are still more than N_2 occupied buffers after the transfer is completed. This scheme can be put into the real system very easily, because the transfer to and from the disk is actually a copy process followed by a deletion process. Now we define :

P_i = probability that system is in state i

= probability there are i occupied buffers.

To simplify notation, we use the following:

$$\lambda_i = \begin{cases} 0 & i<0 \text{ or } i>N. \\ \lambda_i & 0 \le i \le N \end{cases}$$

$$\mu_i = \begin{cases} \mu_i & 0<i\leq N \\ 0 & \text{otherwise} \end{cases}$$

$$\alpha_i = \begin{cases} \alpha & 0\leq i\leq N_1 \\ 0 & \text{otherwise} \end{cases} \qquad 3.4.1$$

$$\beta_i = \begin{cases} \beta & N_2\leq i\leq N \\ 0 & \text{otherwise} \end{cases}$$

and $P_i = 0$ for $i<0$ and $i>N$.

Then the balance equation becomes

$$(\lambda_i + \mu_i + \alpha_i + \beta_i) P_i = \lambda_{i-1}P_{i-1} + \mu_{i+1}P_{i+1} + \alpha_{i-k}P_{i-k} + \beta_{i+k}P_{i+k}$$

for all $0\leq i\leq N$ 3.4.2

P_i can be solved with the normalization constraint, $\Sigma P_i=1$. A special technique is developed to solve this problem without inverting the (N+1)X(N+1) matrix. We will call this technique forward and backward algorithm. This algorithm can also be used to solve a much more complicated problem, as will demonstrate in the next section.

3.4.2. Forward and Backward Algorithm

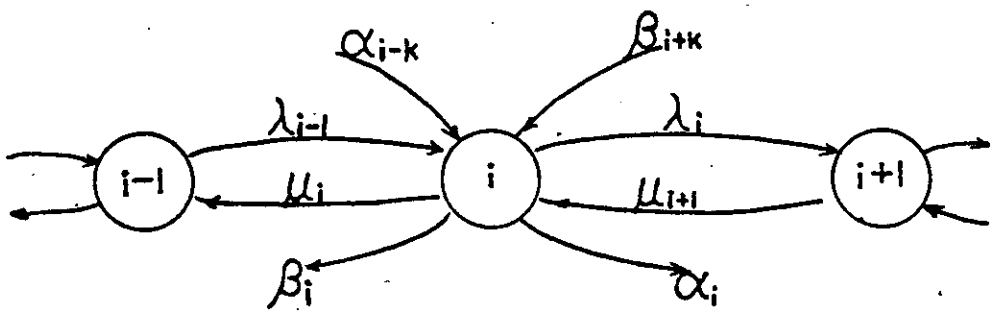
The algorithm is similar to the recursive technique developed by Herzog et al.[Her75]. The recursive technique introduces the substitution

$$P_{i,j} = \sum_{r=1}^{K} C_{i,j}^{r} P_{N,r}$$

for some boundary states $P_{N,r}$, and, from some boundary conditions and the balance equations, $C_{i,j}^{r}$ are calculated. The algorithm has a serious drawback: When the transitions becomes complicated, as they do here, no single set of boundaries can be found to solve the whole system. Here we suggest that two sets of boundary states be chosen instead of one. From the balance equations, the boundary states will meet

somewhere in the state space. By matching those probabilities, we will obtain the key to solve the whole system.

The basic transitions to and from state i are:



At most six of the eight transitions to and from state i will be nonzero.

$$\alpha_i \cdot \beta_{i+k} = \alpha_{i+k} \cdot \beta_i = 0$$

If the boundary chosen is P_0, then all the transitions from states i-1, i, and i-k are known and can be used to calculate state (i+1). The process will stop at i when β_{i+k} is nonzero. The same thing occurs if the boundary is chosen as P_N and the recursive process stops at i when $\alpha_{i-k} \neq 0$. So, instead of one boundary state, there are two, P_0 and P_N. This is the philosphy of the forward and backward algorithm.

Let

$$P_i = a_i P_0 + b_i P_N \qquad 3.4.3$$

then we will have

$$(\lambda_i + \mu_i + \alpha_i + \beta_i)\, a_i\, P_0 + (\lambda_i + \mu_i + \alpha_i + \beta_i)\, b_i\, P_N$$

$$= (\lambda_{i-1}a_{i-1} + \mu_{i+1}a_{i+1} + \alpha_{i-k}a_{i-k} + \beta_{i+k}a_{i+k})\, P_0$$

$$+(\lambda_{i-1}b_{i-1} + \mu_{i+1}b_{i+1} + \alpha_{i-k}b_{i-k} + \beta_{i+k}b_{i+k})\, P_N$$

Set the coefficient of P_0 on both side of the equation equal, and we will have:

$$(\lambda_i + \mu_i + \alpha_i + \beta_i)\, a_i = \lambda_{i-1}a_{i-1} + \mu_{i+1}a_{i+1} + \alpha_{i-k}a_{i-k} + \beta_{i+k}a_{i+k}$$

$$(\lambda_i + \mu_i + \alpha_i + \beta_i)\, b_i = \lambda_{i-1}b_{i-1} + \mu_{i+1}b_{i+1} + \alpha_{i-k}b_{i-k} + \beta_{i+k}b_{i+k}$$

3.4.4

with the boundary conditions that $a_0=1$, $b_0=0$, $a_N=0$, $b_N=1$.

The forward process is

$$X_{i+1} = \mu_{i+1} [(\lambda_i+\mu_i+\alpha_i+\beta_i)X_i - \lambda_{i-1}X_{i-1} - \alpha_{i-k}X_{i-k} - \beta_{i+k}X_{i+k}] \quad 3.4.5$$

and the backward process is

$$X_{i-1} = [(\lambda_i+\mu_i+\alpha_i+\beta_i)X_i - \mu_{i+1}X_{i+1} - \alpha_{i-k}X_{i-k} - \beta_{i+k}X_{i+k}]/\lambda_{i-1} \quad 3.4.6$$

The algorithm calls for using the forward process to calaulate X_i from i=0, 1, 2 , · , · , as far as possible until β_{i+k} is nonzero. The backward process is also used to calculate X_j from j=N, N-1, N-2, · , · , . Where X_i is either a_i or b_i. The two processes will meet, and we will have two different sets of a_j and b_j representing the same P_j, or we will have

$$\begin{aligned} P_j &= a_jP_0 + b_jP_N \\ &= a_j'P_0 + b_j'P_N \end{aligned} \quad 3.4.7$$

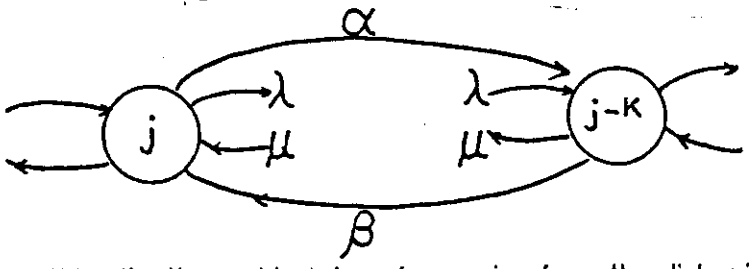
Then we get $P_0 = (b_j-b_j')P_N/(a_j'-a_j)$ and

$$P_i = [(b_j-b_j')/(a_j'-a_j) \cdot a_i + b_i]P_N \quad 3.4.8$$

and

$$P_N = [\sum_{i=0}^{N} \{(b_j-b_j')/(a_j'-a_j) \cdot a_i + b_i\}]^{-1} \quad 3.4.9$$

There is one situation in which the above algorithm does not give a solution is the one in which the following states exist in the system:



i.e., $N_2-N_1 \leq K$. In this situation, a block transfer coming from the disk will automatically generate a block transfer request back to the disk. The data structure of the algorithm are two array A[0:N] and B[0:N].

Algorithm 3.1

[0](initial) Read in values of α, β, λ, μ, N1, N2, N, K. Set A[I]=B[I]=small for I=1,2,$\cdots$,N, where small = -10.0^{-10}. I1=0, I2=N, and A[0]=B[N]=1, A[N]=B[0]=0

[1](forward process) If I1+K≥N2 and A[I1+K]=small then go to [3]; otherwise

X ← right hand side of equation (3.4.5) replace X_i by A[I].

Y ← same as above equation except for B[ ] rather than A[].

I1←I1+1,

[2] if I1=I2 then goto [5]; otherwise A[I1]←X, B[I1]←Y and goto [1].

[3](backward process) If I2-K≥N1 and B[I2-K]=small then goto [1]; otherwise set

X ← right hand side of equation (3.4.6) replace X_i by A[I].

Y ← same as above equation except for B[ ] rather than A[ ].

Set I2←I2-1;

[4] If I1=I2 then goto [5]; otherwise A[I2]←X, B[I2]←Y, goto [3]

[5](calculate the ratio P0 and PN) se← (B[I1]-Y)/(X-A[I1])

[6] Set

$$P[N] \leftarrow \{\sum_{I=0}^{N} (R*A[I]+B[I])\}^{-1}$$

[7] P[I] ←(R*A[I]+B[I])*P[N] for I=0,1,2,$\cdots$, N-1.

Let us review this algorithm in relation to the problem. First we will go to forward process and calculate I1=0,1,2,3, $\cdots$ until I1=N2-K. Then we will switch to the backward process. The backward process will calculate I2=N,N-1,$\cdots$, until for some value, either I1=I2 or ALPHA[I]>0. The latter case implies that I2≤N1+K; then we go back to the forward process. Because the A[I1] and B[I1], which are A[N2-K] and B[N2-K], are already calculated at backward process, the forward process can go on

until I1=I2. After I1=I2, we can use equation (3.4.8) to calculate the ratio of the two boundaries, P0 and PN. Then from the normalization constraint, the exact value of the boundary states can be calculated.

3.4.3. Numerical Results

The secondary storage device is suggested mainly for use with the integrated switch often in overloaded states. So, unlike in Section 3.2, where $\mu_i=i\mu$ is assumed, the service rate of occupied buffers, or the rate at which the occupied buffers are released, is assumed to be:

$$\mu_i\{ \begin{array}{ll} i\mu & i\le c_i \\ c_i\mu & i\ge c_i \end{array} \qquad 3.4.10$$

where c is the number of virtual servers of the system without the free buffer queue, or equal to τ/b of equation (3.3.21). c is also the breaking point of dotted lines of Figure 3.3.1. From this definition, c may not be an integer and may not be equal to any number of physical servers of the system. $c\mu$ is the maximum throughput the queueing network can handle. So, if the number of occupied buffers is less than c, some of the processing power is wasted. Where the utilization factor of the network is defined as:

$$R = \sum_{i=0}^{N} \frac{\min(i,c)}{c} P_i \qquad 3.4.11$$

the system is fully utilized if the number of occupied buffers is greater than c.

Here μ is the b of equation (3.3.21), i.e., $1/\mu$ is the average total service time required for a buffer to pass through the network. Suppose a switch will wait $2/\mu$ time to determine that a packet is blocked and to retransmit it. As we discussed before, $2/\mu$ will be of the same order of magnitude as the transfer time of the disk.

So, $\alpha=\beta=\mu/2$ is assumed in most of the numerical results, i.e., the extra delay required to retransmit the blocked packet is the same as that required to put the packet onto a disk.

Blocking probability and the utilization factor are the two criteria used here. Figure 3.4.1 shows a typical relationship between blocking probability and the utilization factor when the system processing capability changes. It also shows the advantage of the secondary storage device. Figure 3.4.2 shows the effect of c: The number of virtual servers of the network, changes if the traffic intensity, λ/μ, is kept constant. Figure 3.4.3 and 3.4.4 show the effect of the choice of N_1 and N_2: In overloaded states, $\lambda>c\mu$, the choice is rather unimportant, while it is quite critical in under-loaded states, $\lambda<c\mu$. Figure 3.4.5 shows the effect of the disk transfer rate: The blocking probability decreases linearly as the disk transfer rate increases. For $\lambda=c\mu$, the blocking probability almost vanishes when $\alpha=\beta=\mu$. For overloaded states, $\lambda=1.5*c\mu$, the blocking probability becomes half for a disk with transfer rate $\alpha=\mu$, where $1/\mu$ in general will be in the order of 30 milliseconds or more. So even a moderate speed disk can improve the system behavior greatly. Figure 3.4.6 shows the effect of using the Erlang(α,k) disk transfer time instead of the exponential transfer time, where Erlang(α,k) distribution has a mean of $1/\alpha$ and a variance of α^2/k. In this figure, Erlang distributions of the same mean, but with degree one and 10, are compared. Although the distribution looks a little different, the blocking probability and utilization factor differ from each other very little.

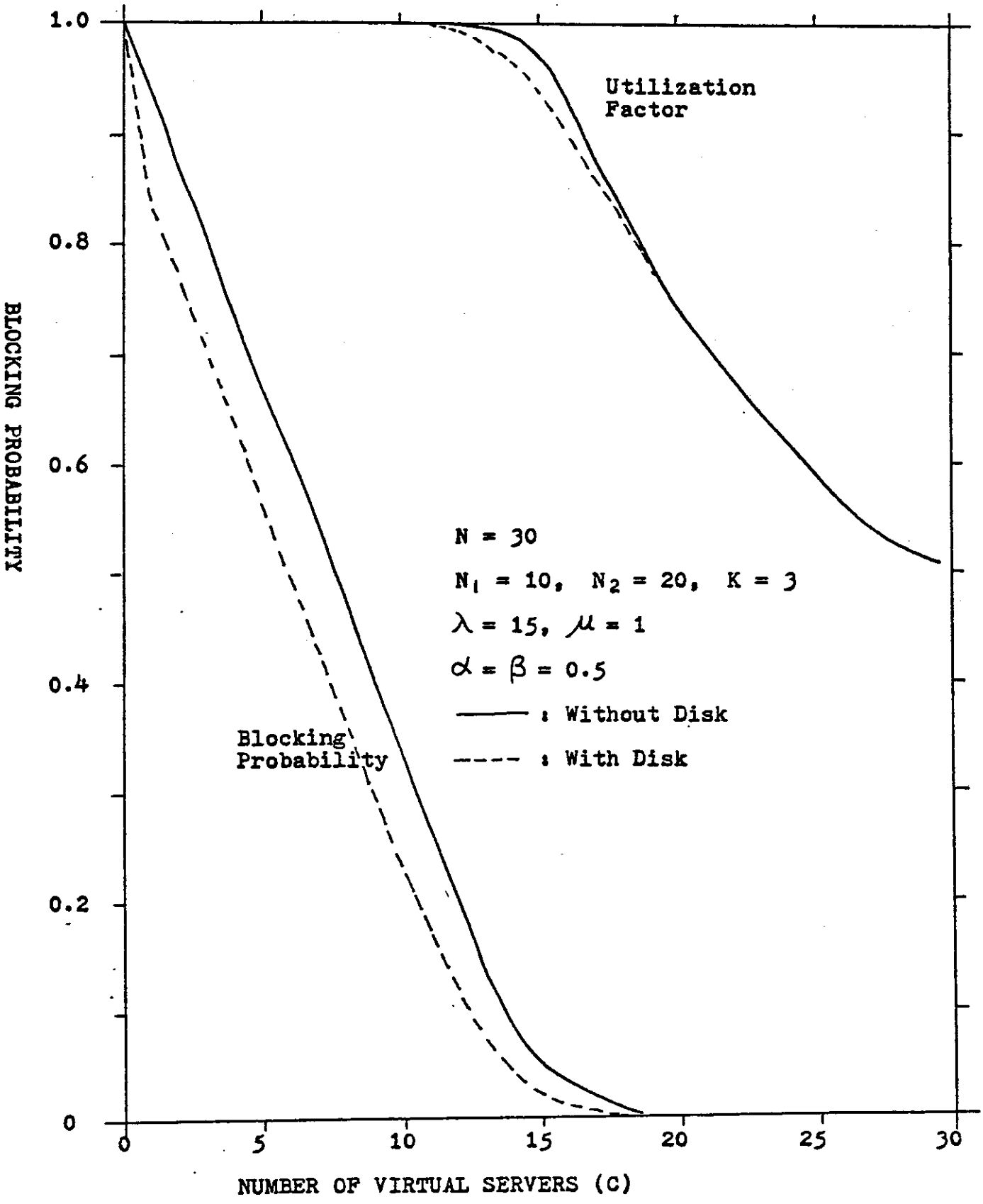


FIGURE 3.4.1 COMPARISON OF SYSTEM WITH OR WITHOUT DISK

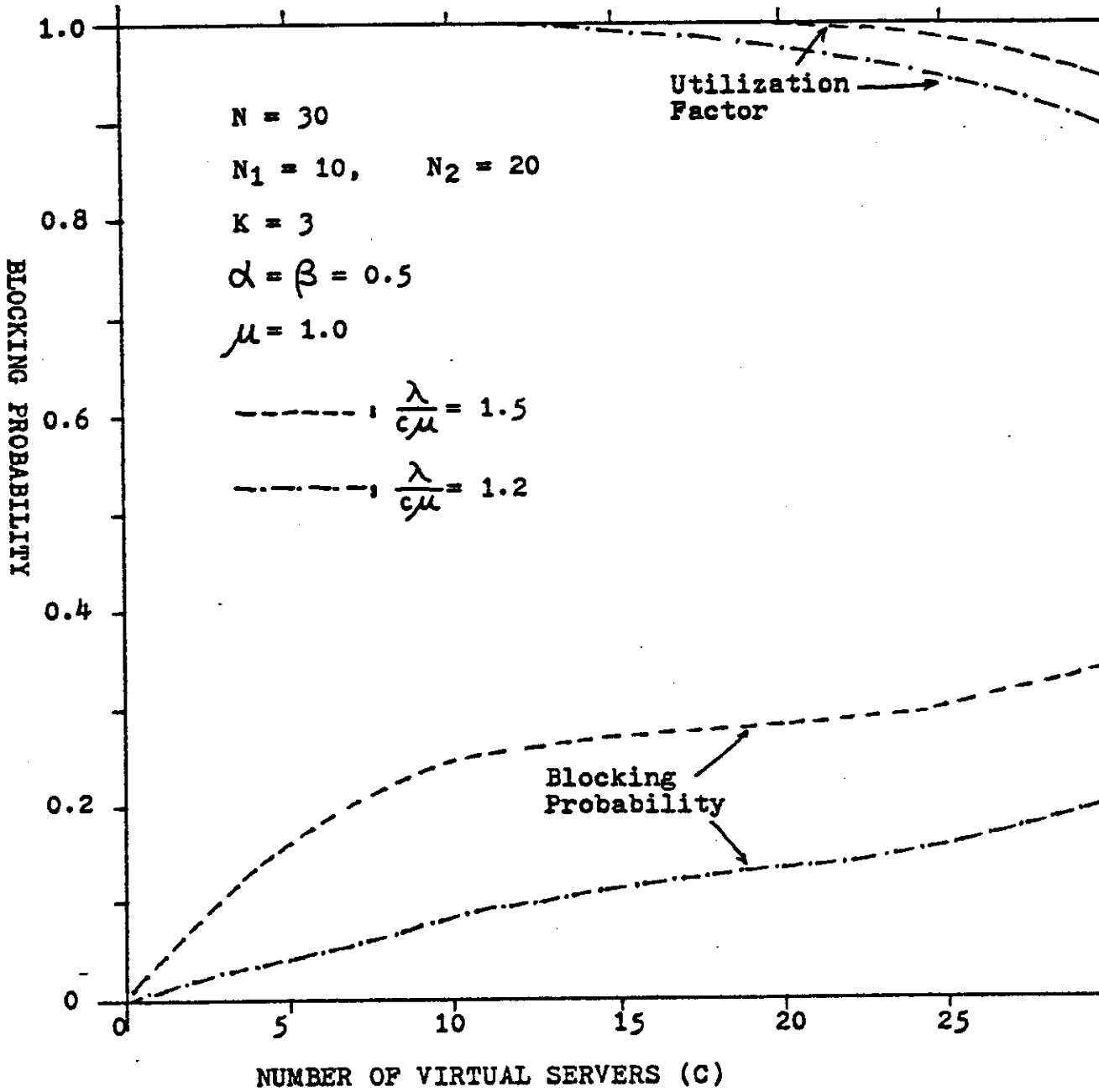


FIGURE 3.4.2 EFFECTS OF DIFFERENT VIRTUAL SERVERS

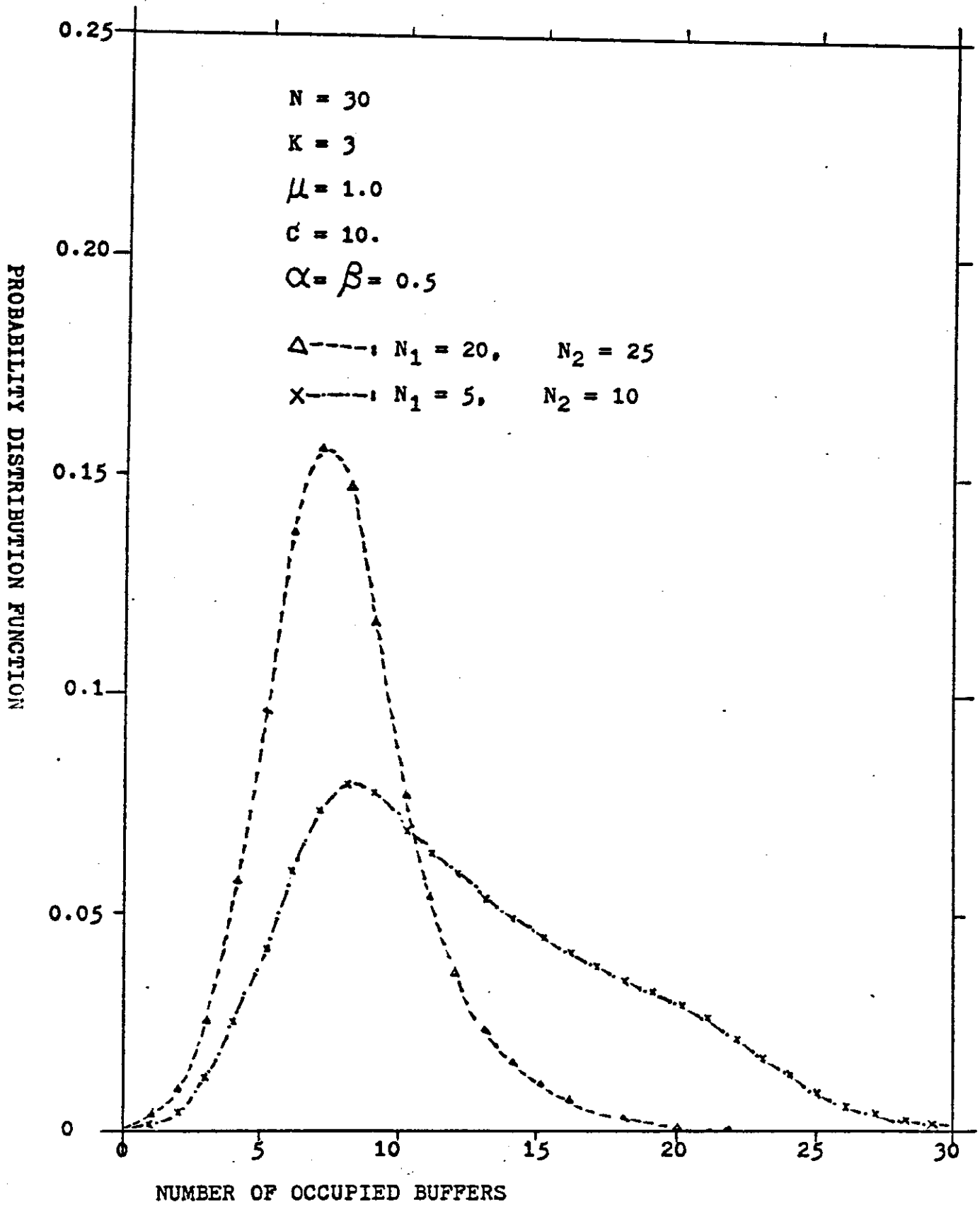


FIGURE 3.4.3 EFFECT OF N_1, N_2

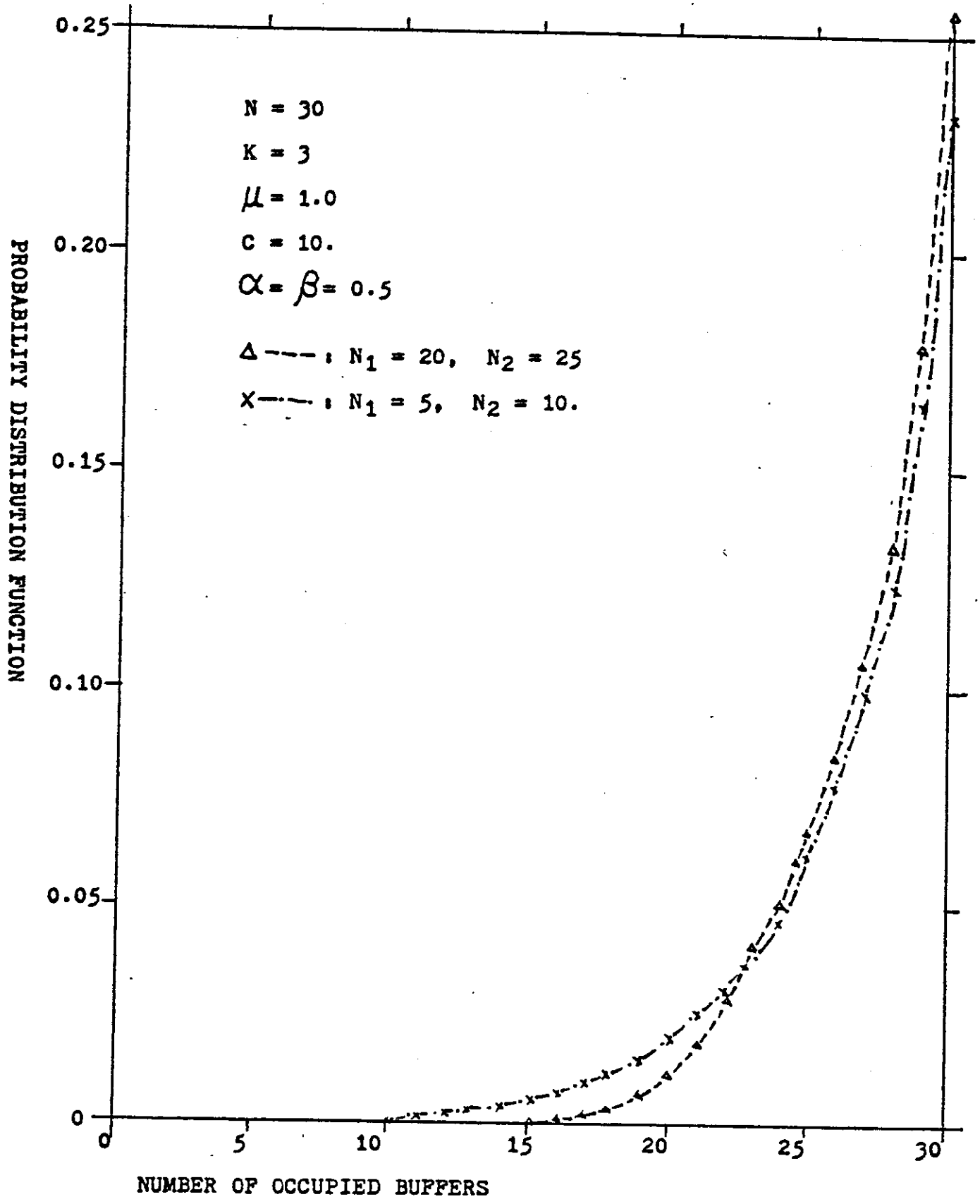


FIGURE 3.4.4 EFFECT OF N_1, N_2 IN HEAVY LOAD CONDITION

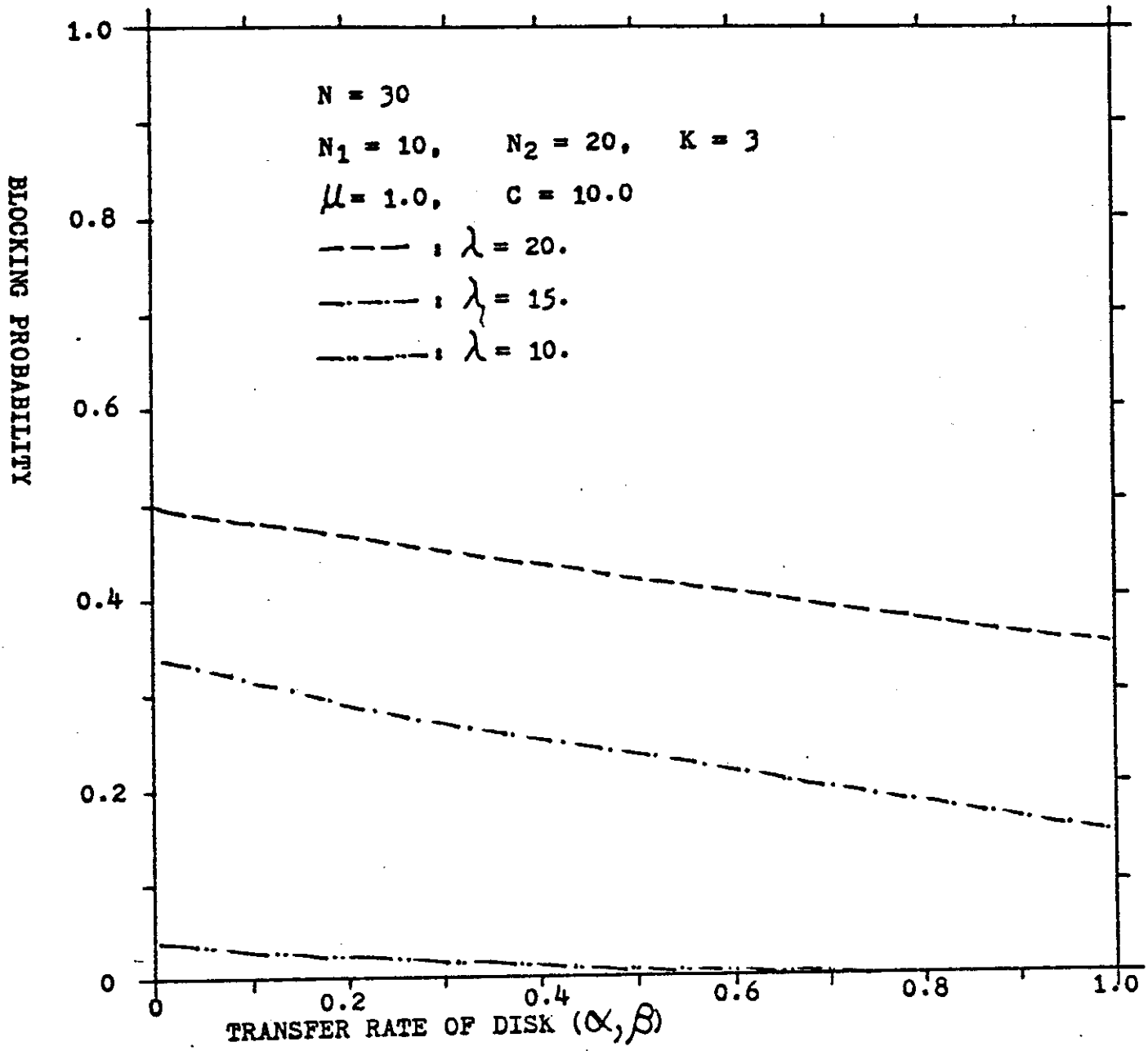


FIGURE 3.4.5 EFFECT OF DIFFERENT DISK RATE

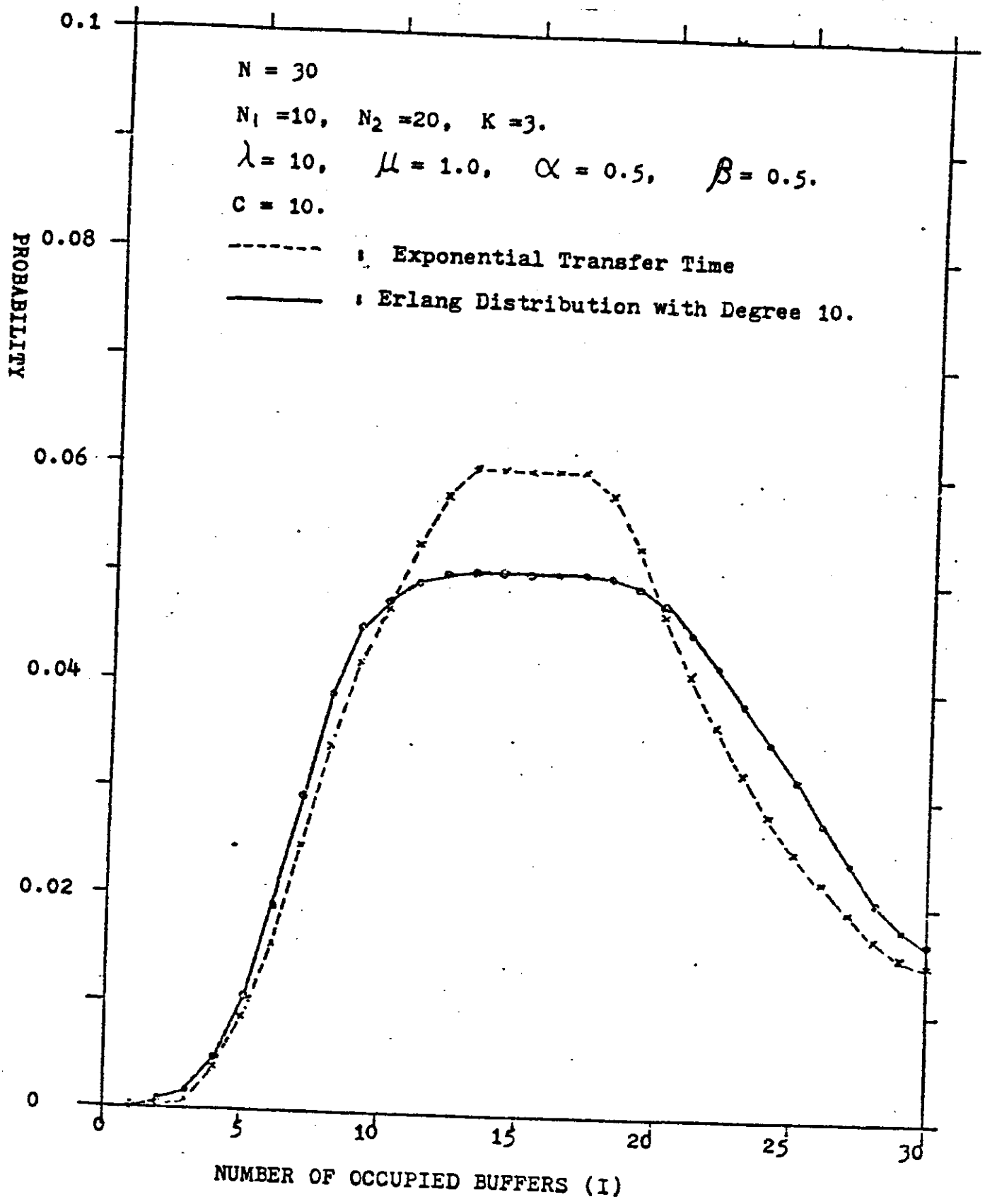


FIGURE 3.4.6 EFFECT OF ERLANG DISTRIBUTION TRANSFER TIME

3.5. Conclusion

Both first-fit algorithm and dynamic assignment of fixed-size buffers sacrifice the larger size packets too much in a heavily loaded system. But both give a superior memory management than other schemes in a lightly loaded system. The proper choice of a management schemes depends on the real environment of the communication network. Also the priority system discussed in section 3.2 is suggested. A small sacrifice of the low priority job can decrease greatly the blocking probability, and shorten the response time of high priority jobs. For uniformly distributed packet size, the choice of the buffer size is not critical, while the method of distributing the memory among different size buffers is important. A secondary storage device might well be worth the trouble. Even a relatively slow disk can improve the network performance, especially if there is a high probability that integrated switch will enter the overloaded states.

CHAPTER 4 SENET Network Design

4.1. Introduction

The special frame structure of the SENET network brings up the two special problems: the channel capacity assignment problem(CA) and the frame skew assignment problem (FA). The CA problem is solved once and for all for other communication networks at the time of their design stage. However, the SENET network will dynamically assign the channel capacity to data traffic under different loading of voice traffic, and therefore the CA problem also has to be solved dynamically. The FA problem exists only for the special frame structure of the SENET network. As mentioned in Chapter 3, the information contained in a frame can only be processed after the whole frame is received accurately, and no information in a frame can be changed once the switch has begun transmitting the frame. Different frame skew assignments may differ by from several milliseconds to one frame period at one switching node. This delay is essential for the voice slots which have to pass several nodes from source to destination.

In Section 2, general design variables, performance measures and a formulation of the problem of designing a SENET computer-communication network are stated. In Section 3, the capacity assignment problem is viewed with the small frame assumption. It becomes the general assignment problem that [Ger73] among others has solved. Then the coordinate descent method and Newton's method are used to solve the general case. A simple example is also given showing the capacity assignments of different criteria and constraints. In Section 4, the skew assignment problem is solved

as a MILP(Mixed Integer Linear Problem). In Section 5, a heuristic algorithm is given for the skew assignment problem and the results are compared with those of Section 4.

4.2. The Model

There will be three sub-sections: variables, performance criteria, and the formulations of the problem.

4.2.1. Variables

Here we list the variables that are to be considered in this chapter. The list is by no means complete and a more sophisticated design might involve additional variables(such as: reliability, priority for voice and data, etc.)

Topology: We consider a directed network [N:A], where N is the set of nodes(of cardinality n), and A is the set of directed arcs(of cardinality m). We shall use N_i to indicate the ith node and A_{ij} to indicate the directed arc from N_i to N_j. For convenience of notation, we also index the links as directed arcs with subscript to refer to the jth link as A_j, and index the node N_{ij} as the node at which the ith link is connected to jth link. Sets [N:A] are the topology of the communication network and are assumed to be fixed over the whole chapter.

Channels: To each link is assigned a capacity c_i which is the bandwidth for data traffic; such capacity will be only considered as a continuous variable. Also, for a given link i there is a cost D_i associated with value of capacity:

$$D_i = d_i(c_i) \qquad 4.2.1$$

Which may be the real cost of the links or just a reflection of voice loading on the link. For the linear cost situation we will use d as the cost vector, in which d_i, the ith element of d, is the capacity-cost of ith link.

Traffic Matrix: Let λ_{ij} [packets/sec] be the required average rate of transmission of

data packets from link i to link j. If link i is not directly connected to link j then $\lambda_{ij}=0$. $\lambda_{0,j}$ is the traffic generated by the node where link j begins. λ_{i0} is the traffic destined for the node where link i ends. The routing policy of the packet is not considered here.

The channel flow λ_j is the traffic on link j.

$$\lambda_j = \sum_{i=0}^{m} \lambda_{ij} = \sum_{i=0}^{m} \lambda_{ji} \qquad 4.2.2$$

Let $1/\mu$ [bits/packet] be the average packet size. λ_j has to be less than $c_j\mu$ for all the links, otherwise the queue length of link j will be infinite.

The throughput λ of the whole network is defined as the summation of all channel flow.

$$\lambda = \sum_{i=1}^{m} \lambda_i = \sum_{i=1}^{m}\sum_{j=0}^{m} \lambda_{ij} \qquad 4.2.3$$

Frame Skew: Let s_j be the frame skew of link j. s_j is defined to be greater than or equal to zero and less than F, the frame period. All the frame skews are relative to each other, so there is one degree of freedom of skew assignment. Usually we set $s_1=0$, which means the first link is the standard link. The skew assignment will only affect the delay of traffic being transferred it will not affect the traffic when it is first generated and when it ends at the node.

Other variables: The storage available in the node could also be considered as a design variable and associated with a cost. However, in this chapter no such consideration is made.

Priority can be assigned to packets according to their nature (acknowledgement, control packet intrinsic priority, etc.) or weighed by their length. The possibility of dividing the packet into priority classes(with the number of classes and boundaries as design variables) would probably affect the design of the network; however, we do not consider such a feature and assume that all queues are managed on a FIFO basis.

4.2.2. Performance Measures

Several criteria can be adopted to measure the performance of a communication network. The lists here are the performance measures most commonly used.

Average Message Delay T

Recall, from [Kle70], that the general expression of the message delay T is of the following form:

$$T = \frac{1}{\lambda}\sum_{j=1}^{m}\sum_{i=0}^{m}\lambda_{ij}[T_{ij}(s_i,s_j,p_i,n_{ij})+p_i] \qquad 4.2.4$$

where λ = the total throughput [packets/sec]

s_i = frame skew of ith link.

p_i = propagation delay in link i [sec/messg]

n_{ij} = processing time in the node connecting link i to link j.

T_{ij} = delay on the node connecting link i to link j [sec/packets].

T_{ij} consists of the node processing time, queueing time for link j and the waiting for the frame to be transmitted. For the special frame structure of the system, T_{ij} has the following form:

$$T_{ij} = s_{ij}+r_{ij}(s_{ij})\cdot F \qquad \text{for } i\neq 0 \qquad 4.2.5$$

where s_{ij} is the mis-matched delay of link j from link i.

$$s_{ij} = s_j-(s_i+p_i+n_{ij}) \mod F$$

is the time left for data packets to be transmitted after they have arrived at the node and had the necessary processing.

Because the number of frames is a non-negative integer random number depending on s_{ij}, r_{ij} is the number of frames the data packet has to wait until it is

transmitted. For λ_{0j} traffic, s_{0j} is assumed to be uniformly distributed in the interval [0,F) and r_{ij} is half a frame period.

A queue for link j is formed. The input is a joint of streams λ_{ij} for all i, each of the streams comes in at the discrete time $-s_{ij}+k\cdot F$ for all integers k. The average T_{ij} becomes

$$T_{ij} = s_{ij} + \sum_{k=0}^{\infty} \text{Prob}[(k-1)\cdot F \le W - s_{ij} < k\cdot F]\cdot kF \qquad 4.2.6$$

where W is the waiting time at the queue of link j.

In this chapter, we will make the following assumptions in different situation.

(i) Independent Assumption: We are using the assumption made by Kleinrock[Kle72] that the independence between the arrival time and the length of a data packet can be interpreted as that the packet enters the network from an external source. With exponentially distributed packet length and no acknowledgement traffic [Kle70], the queue is modeled as a M/M/1 system.

$$\text{Prob}[r_{ij}(s_{ij}) = k] = \text{Prob}[(k-1)\cdot F \le W - s_{ij} < k\cdot F] \qquad 4.2.7$$

$$= \int_{(k-1)F+s_{ij}}^{kF+s_{ij}} dF(W) \qquad 4.2.8$$

where

$$dF(W) = (\mu c_j - \lambda_j)\exp[-(\mu c_j - \lambda_j)W]\, dW \qquad 4.2.9$$

So

$$\bar{r}_{ij} = \sum_{k=1}^{\infty} k\cdot \text{Prob}[(k-1)\cdot F \le W - s_{ij} < k\cdot F]$$

$$= \sum_{k=1}^{\infty} k\cdot \exp\{-[(k-1)F+s_{ij}](\mu c_j - \lambda_j)\} - k\cdot \exp\{-(kF+s_{ij})(\mu c_j - \lambda_j)\}$$

$$= \exp[-(\mu c_j - \lambda_j)s_{ij}] \sum_{k=1}^{\infty} \exp[-k(\mu c_j - \lambda_j)F]$$

$$= \frac{\exp[-(\mu c_j - \lambda_j)s_{ij}]}{1-\exp[-(\mu c_j - \lambda_j)F]} \qquad 4.2.10$$

$$\bar{T}_{ij} = s_{ij} + \bar{r}_{ij} \cdot F \qquad i>0 \qquad 4.2.11$$

for

$$\bar{r}_{0j} = F/2 + F/(\mu c_j - \lambda_j) \qquad 4.2.12$$

(ii) Deterministic Assumption. The packet size and the traffic flow are deterministic. The voice slots can be considered as a deterministic packet flow in the network. This is also the limiting case that the frame period is large compared to the mean service time of a link for a data packet. By the law of large numbers, the standard deviation of the summation of n independent random numbers is only 1/n of each of the individuals. In the frame structure, all the packets coming in a frame period are packed and transmitted together. The variance of waiting time decreases as the frame size increases, and $r_{ij}>0$ is negligble compared to s_{ij}. We have

$$\bar{T}_{ij} = s_{ij}.$$

Mean-kth-Power Delay T(k)

In the paper [Mei72] it was observed that, in minimizing T, a wide variation was allowed among the delays T_{ij}. In order to take into account such variation, the following alternative performance measure was proposed:

$$T^{(k)} = \left[\frac{1}{\lambda} \sum_{j=1}^{m} \sum_{i=0}^{m} \lambda_{ij} \, T_{ij}^{k} \right]^{1/k} \qquad 4.2.13$$

where $k \geq 1$. If k is large enough, the variation among packet delays is considerably reduced at the expense of a higher average delay. [Mei72]

Maximum Average Delay Tmax

Maximum average delay is the special case of mean-kth-power when k goes to infinite, it becomes the so called "mini-max criterion". This performance measure is used when the guaranteed maximum delay can not exceed some prefixed value. In a communication network, especially for voice and interactive data, this performance measure is often used at the expense of a higher average delay.

$$T_{max} = \max_{\text{over all } i,j} \{ T_{ij}i \}$$

Cost

The total cost of the communication is assumed to be:

$$D = \sum_{i=1}^{m} d_i(c_i)$$

where the function $d_i(\cdot)$ may be the real physical cost of the communication links or just a reflection of the blocking probability of voice calls. $d_i(\cdot)$ may be linear or concave, depending on the characteristics of the link.

There are other important criteria such as the throughput and the reliability of the system, which we will not discuss here.

4.2.3. Design Problems

(i)Optimum Assignment of Capacity (CA problem)

given : topology, throughput, routing, frame skew

objective : minimize T

design variables : capacities.

constraints : $D = \sum_{i=1}^{m} d_i(c_i) \leq D_{max}$ 4.2.14

$\mu c_j \geq \lambda_j$ for all j

(ii)Optimum Assignment of Frame Skew (SA problem)

given : topology, throughput, routing, capacity

objective : minimize T

design variables : frame skew.

constraints : $\mu c_j > \lambda_j$ for all j 4.2.15

4.3. Capacity Assignment Problem

given : topology, thruput, routing, frame skew

minimize T

design variables : capacities.

constraints : $D = \sum_{i=1}^{m} d_i(c_i) \le D_{max}$ 4.3.1

$\mu c_j \ge \lambda_j$

The independent assumption of [Kle72] is used. T_{ij} is assumed as follows:

$$\bar{T}_{ij} = s_{ij} + \frac{\exp[-(\mu c_j-\lambda_j)s_{ij}]}{1-\exp[-(\mu c_j-\lambda_j)F]} \cdot F \qquad 4.3.2$$

where $s_{ij} = s_j-(s_i+p_i+n_{ij}) \mod F$, and $0 \le s_{ij} < F$

4.3.1. Small Frame Capacity Assignment

Here the CA problem is solved for the small frame assumption. At the end of this section we will explain the physical meaning of this assumption. It will be clear then that this assumption may apply to some systems whose the frame sizes are not small.

The small frame assumption is that the $(\mu c_j-\lambda_j)F$ is very small, all the nonlinear term of the Taylor Series Expansion of equation (4.3.2) are neglected.

$$\bar{T}_{ij} = s_{ij} + \frac{1-[\mu c_j-\lambda_j]s_{ij}}{[\mu c_j-\lambda_j]F} F$$

$$= s_{ij} + \frac{1}{\mu c_j-\lambda_j} - s_{ij}$$

$$= \frac{1}{\mu c_j-\lambda_j} \qquad 4.3.3$$

This is a well-known formula for ordinary communication networks [Kle72]. Each

link is modeled as a M/M/1 queue with exponential service time and Poisson input. The service rate of a link is c_j [bits/sec] and the input rate of data packets is λ_j [packets/sec]. The average packet length is $1/\mu$ [bits/packets] and the average waiting time is $\mu\bar{T}_{ij}$ [seconds per packet].

The problem becomes:

given : topology [N:A], data traffic routing λ_{ij}, and frame skew assignment s_{ij}.

$$\underset{\text{over } c}{\text{minimize}} : T = \{\frac{1}{\lambda}\cdot\sum_{j=1}^{m}\lambda_j\cdot[\frac{1}{\mu c_j-\lambda_j}]^k\}^{1/k} \qquad 4.3.4$$

$$\text{subject to} : \sum_{i=1}^{m} d_j(c_j) \le D_{max}$$

$$\mu c_j \ge \lambda_j \qquad \text{for all } j$$

This is the general mean-kth-power delay criterion [Mei72].

For the monotonic property of function $x^{1/k}$, to minimizing x^k is equivalent to minimizing x under the same set of constraints. So the problem can be written as:

$$\underset{\text{over } c}{\text{minimize}} : T = \frac{1}{\lambda}\sum_{j=1}^{m}\lambda_j[\frac{1}{\mu c_j-\lambda_j}]^k \qquad 4.3.5$$

$$\text{subject to} : \sum_{i=1}^{m} d_j(c_j) \le D_{max}$$

$$\mu c_j \ge \lambda_j \qquad \text{for all } j$$

First the linear cost-capacity function is considered. The objective function is convex because it is a sum of convex terms and the set of possible c is also convex, therefore a local minimum is also a global minimum [Had64]. This problem can be solved using the Lagrange Multiplier.

The Lagrangian L for the problem is

$$L = T+\beta(D-D_{max}) \qquad 4.3.6$$

$$= \frac{1}{\lambda}\sum_{j=1}^{m}[\frac{1}{\mu c_j-\lambda_j}]^k + \beta(\sum_{j=1}^{m} d_j c_j - D_{max}) \qquad 4.3.7$$

where β is the Lagrangian multiplier.

By differentiating the equation with respect to c_j, we obtain

$$\frac{\partial}{\partial c_j} L = \frac{\lambda_j}{\lambda}\left[\frac{1}{\mu c_j - \lambda_j}\right]^{(k-1)} \cdot \frac{(-1)}{(\mu c_j - \lambda)_j^2} + \beta d_j \qquad 4.3.8$$

By setting the partial derivative to zero we obtain the optimum expression for c_j

$$\mu c_j = \lambda_j + \alpha(\lambda_j/d_j)^{1/(k+1)} \qquad 4.3.9$$

where α is a parameter independent of j. Computing α by satisfying the cost constraint obtain:

$$\alpha = (D_{max} - \sum_{j=1}^{m} d_j\lambda_j)/[\sum_{j=1}^{m} (\lambda_j/d_j)^{1/(k+1)}] \qquad 4.3.10$$

This expression of μc_j is first derived by [Mei72]. When k=1, the problem is to minimize the average delay of the network, and the optimum assignment of c_j is so called "square root" assignment [Kle72]. For the min-max criterion, $k\to\infty$, c_j evenly distributes the residue capacity $(D_{max} - \sum d_j\lambda_j)$.

$$\mu c_j^{(\infty)} = \lambda_j + \alpha^{(\infty)} = \lambda_j + (D_{max} - \sum d_j\lambda_j)/m \qquad 4.3.11$$

Concave Cost-Capacity Function

For the concave cost-capacity function, the algorithm to find the local minimum can be constructed similarly to the algorithm developed by [Ger73] which solved the special case of k=1. No algorithm to find the global minimum existed. The algorithm follows, the detailed proof of existence and convergence is cited in [Fra72] and [Ger73].

Let us assume that $d_i(c_i)$ is a concave, nondecreasing function of c_i, for i=1, 2, 3, . , . , N. (see Figure 4.3.1)

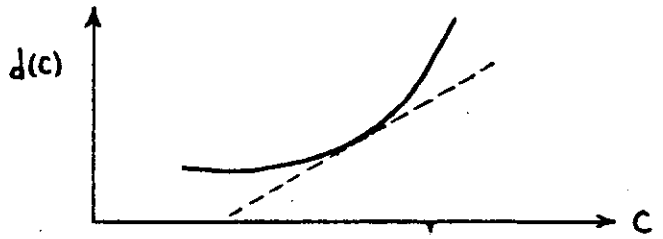


FIGURE 4.3.1 CONCAVE NONDECREASING COST FUNCTION

The inspection of the cost constraint of equation (4.3.1) shows that the set of feasible c_i's is not convex. Therefore, there exist in general, several local minima. If we assume that all functions $d_i(c_i)$ are continuous and differentiable for $c_i > f_i$, then the local minimum is characterized by the following properties:

Property 4.1

If $\bar{c}$ is a local minimum for the concave problem, then it is also a global minimum for the problem with cost-capacity curves linearized around $\bar{c}$. (see Figure 4.3.2)

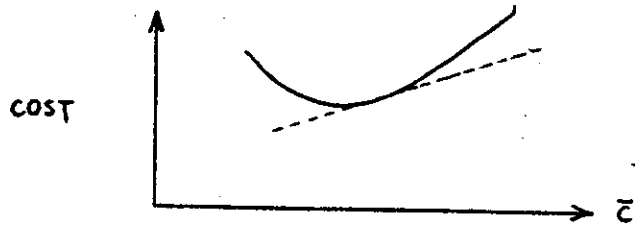


FIGURE 4.3.2 LOCAL MINIMUM

The above property is an immediate consequence of the fact that the set of feasible moves ∂c around $\bar{c}$ is the same for both concave and linearized problems.

Property 4.2

If $c^{(n)}$ is a feasible assignment, and $c^{(n+1)}$ is the solution of the problem linearized around $c^{(n)}$, then

$$T(c^{(n+1)}) \leq T(c^{(n)}) \qquad 4.3.12$$

$$D_{concave}(c^{(n+1)}) \leq D_{linear}(c^{(n+1)}) \leq D_{max} \qquad 4.3.13$$

where $D_{concave}(c^{(n+1)})$ is the cost computed on the concave curves and $D_{linear}(c^{(n+1)})$ is the cost computed on the linearized curves of $c^{(n)}$.

Property (4.1) and (4.2) lead to the following algorithm for the determination of local minima.

Algorithm 4.1 Let $c^{(0)}$ be a starting feasible assignment.

[0] (initial) Let $n=0$, $T_0=\infty$

[1] Compute $c^{(n+1)}$=solution of the problem linearized around $c^{(n)}$ and compute $T_{n+1}=T(c^{(n+1)})$.

[2] If $|T_{n+1}-T_n|<\epsilon$, where ϵ is a proper positive tolerance, stop; $c^{(n+1)}$ is a local minimum within the tolerance, otherwise, let $n=n+1$ and go to [1].

There is an important case of the concave cost-capacity funtion in which the local minimum is unique and coincides with the global minimum. Such a case is known as the "power law cost function" [Kle70] and the cost function is given by:

$$d_i(c_i) = d_i c_i^{\alpha} + d_{i0} \qquad 4.3.14$$

where $0 \leq \alpha \leq 1$

The "power law cost" case has been discussed extensively by Kleinrock and a proof of uniqueness of the local minima can be found be in [Kle70].

Now we will discuss the physical meaning of the small frame assumption. The first order Taylor series expansion will be a good approximation if $(1-\lambda_j/\mu c_j)\mu c_j F$ is small for all j. There are two situations for which this condition will hold. One is when $\mu c_j F$ is very small which is why we call it the small frame assumption. $c_{ij}F$, which is the number of bits for data traffic during a frame period is small. This situation may occur on a slow speed link sub-network or on a network with heavily loaded voice traffic

that little capacity is left for data traffic. The other possibility is when $(1-\lambda_j/\mu c_j)$ is small for all j. This is a heavily loaded network for data traffic. The average queue length of links is relatively large and the nature of the frame structure loses its property of discreterness.

If either of the above situations happens, the small frame assumption applies and the network becomes the ordinary packet switching communication network.

4.3.2. General Capacity Assignment

The small frame condition is dropped here. First the linear cost-capacity condition is considered. We have the problem

given : s_{ij}, λ_{ij}, d_i.

$$\underset{\text{over } c}{\text{minimize}} : T = \sum_{j=1}^{m} \sum_{i=0}^{m} \frac{\lambda_{ij}}{\lambda} \left[\frac{\exp[-(\mu c_j-\lambda_j)s_{ij}]}{1-\exp[-(\mu c_j-\lambda_j)F]}\right]^k \qquad 4.3.15$$

$$\text{subject to} : c_j \geq \lambda_j$$

$$\sum_{j=1}^{m} d_j c_j \leq D_{max}$$

The objective T is also convex because it is a sum of convex terms and the set of feasible c's is convex, therefore a local minimum is also a global minimum [Had64]. We first ignore the constraint $\mu c_j \geq \lambda_j$ and verify that the solution satisfies it a posteriori. We also notice that the optimum solution satisfies the cost constraints at equality and therefore regard it as an equality constraint. Now the problem becomes:

$$\underset{\text{over } c}{\text{minimize}}\ T(c) \qquad 4.3.16$$

$$\text{subject to} \sum_{j=1}^{m} d_j c_j = D_{max}$$

Instead of minimizing the above problem, the following unconstrained problem is minimized.

$$\text{minimize } f(c|\beta) = T(c)+\beta\left(\sum_{j=1}^{m} d_j c_j - D_{max}\right)^2 \qquad 4.3.17$$

Let $\{\beta_h\}$, h=1,2,$\cdots$, be a sequence tending to infinity such that for each h, $\beta_h>0$ and $\beta_{h+1}>\beta_h$. Then for each h solve the problem : minimizing $f(c|\beta_h)$ and obtaining a optimum solution $c^{(h)}$. By the theorem of section 12.1 of [Lue73] we know that any limit point of $\{c^{(h)}\}$ is a solution of the origin problem. Because the local minimum is also a global minimum there is only one limit point of $\{c^{(h)}\}$. And for each h such that $\sum_{j=1}^{m} d_j c_j^{(h)} = D_{max}$, then $c^{(h)}=c^{(h+1)}$ and therefore the optimum solution has been found. The algorithm is as followed:

Algorithm 4.2

[0] (initial) set n=0, solve for $f(c|\beta_0)$ by some global convergence algorithm.

[1] If $\sum_{j=1}^{m} d_j c_j^{(h)} = D_{max}$, stop; $c^{(h)}$ is the optimal solution, otherwise $\beta_{h+1}=\alpha\beta_h$.

[2] Solve for $f(c^{(h+1)}|\beta_{h+1})$ by some fast locally convergent algorithm. go to [1].

If we choose too large a β_0 to begin with, the initial solution will converge very slowly, while too small a β_0 will result in a lot of iterations. Here we propose two different algorithms for [0] and [2]. For h=0 the property of global convergence is very important, we have to find some thing to begin with. The Coordinate Descent Method is chosen for its well-known global convergence property. But for h>0, the approximate solution is known, so the local convergence rate is the criterion with which to choose the algorithm. Newton's method for m-dimension is chosen. Newton's method has convergence rate 2 but has to inverse a matrix of dimension m. In this case, there exists a neat analytic form so that the inverse is quickly obtained, and we have a very fast algorithm.

Coordinate Descent Methods

Instead of solving the m dimension problem at once, we solve the problem one dimension at a time.

$$\underset{\text{over } c_j}{\text{minimize}} \; f(c_1, c_2, \ldots c_m) \qquad 4.3.18$$

The cyclic coordinate descent algorithm minimizes f cyclically with respect to the coordinate variables. Thus c_1 is changed first, then c_2, and so forth through c_m. The process is then repeated starting with c_1 again. Although no proof can be given for the global convergence of Newton's method. Newton's method of one dimension for this problem can be proven to be globally converge. For coordinate c_j, $f_j(c_j)$ is the object function. Because $f_j(c_j)$ is convex, $f_j''(c_j)<0$, the local minimum is also the global minimum and there is at most one local minimum point. $f_j(c_j)$ is continuous and its first derivative is also continuous for all $\mu c_j>\lambda_j$. $f_j(c_j)$ is negative at $\mu c_j=\lambda_j$ and positive at $c_j=\infty$, so the stationary point, $f_j'(c_j)=0$, exists. Because of its convex property, $f_j(c_j)$ has a local minimum. In the second derivative of $f_j(c_j)$ with respect to c_j every term is positive and is a monotonic decreasing function of c_j. It is easy to prove that Newton's method is globally and locally convergent for such a function. The iterative formula is as follows:

$$c_j^{(n+1)} \leftarrow c_j^{(n)} - f_j'(c_j^{(n)})/f_j''(c_j^{(n)}) \qquad 4.3.19$$

By the theorem of the coordinate descent method, if every coordinate is globally convergent then the whole problem is also globally convergent.

Newton's Method

Newton's method for m-dimension has a local convergent rate of 2, although it may not be globally convergent. For this problem, it does diverge at points far away from the optimum points. The order of convergence is defined as the supremum of the non-negative numbers p satisfying

$$0 \le \underset{n\to\infty}{\overline{\text{limit}}} \frac{|r_{n+1}-r^*|}{|r_n-r^*|^p} < \infty \qquad 4.3.20$$

for sequence $\{r_n\}$ converge to r^*.

Before explaining the algorithm, let us introduce some notation: For a scalar function f(c) of vector c, the gradient of f(c), $\nabla f(c)$, is a vector whose ith element is the partial derivative of f(c) with respect to c_j. The Hessian matrix, F(c), is defined with the (i,j) element as the secondary partial derivative of f(c) with respect to c_i and c_j. The iterative formula of Newton's method of m-dimension is

$$c^{(n+1)} = c^{(n)} - [F(c^{(n)})]^{-1}\nabla f(c^{(n)}) \qquad 4.3.21$$

f(c) is the $f(c|\beta)$ of equation (4.3.17), it is easy to see that f(c) has second partial derivatives. If c is near the minimum point, F(c) is positive definite and the method is well defined. Now we will prove that F(c) is positive definite for every point. F(c) has a very good property that it can be represented as a diagonal matrix G(c) plus $2*\beta dd'$. Where $G_i(c)$, the (i,i)th element of the diagonal matrix G(c), is equal to the second derivative of T(c) with respect to c_i and is greater than zero for all c_i.

$$F(c) = G(c)+2\beta_k dd'. \qquad 4.3.22$$

where d is the cost-capacity vector, and d' is its transpose. By definition of positive definite.

$$\begin{aligned} x'F(c)x &= x'G(c)x + 2\beta_h(x'd)^2 \\ &= \sum_{aj=1}^{m} x_j^2 G_j(c) + 2\beta_k(x'd)^2 \\ &\ge 0 \qquad \text{for all c.} \end{aligned} \qquad 4.3.23$$

The equality holds only for x=0, by the definition that F(c) is positive definite for all c. Because F(c) can be represent by this form, F(c) can be inversed very easily.

$$\begin{aligned} [F(c)]^{-1} &= [G(c) + 2\beta_h dd']^{-1} \\ &= G^{-1}(c)[I + 2\beta_h G^{-1}(c)dd']^{-1} \end{aligned}$$

$$= G^{-1}(c)[I - 2\beta_h G^{-1}(c)dd'/(1+2\beta_h d'G^{-1}(c)d)] \qquad 4.3.24$$

Here we use the identity formula

$$[I + xy']^{-1} = I - \frac{xy'}{1+x'y} \qquad 4.3.25$$

Concave cost-capacity condition

The constraint is now $\sum_{i=1}^{m} d_i(c_i) \le D_{max}$, and the unconstrained optimum problem becomes:

$$\underset{\text{over } c}{\text{minimize}}\ f(c|\beta_h) = T(c)+\beta_h\left(\sum_{j=1}^{m} d_j(c_j)-D_{max}\right)^2 \qquad 4.3.26$$

The Hessian matrix now has a more complicated second term. Let D(c) be the second term of F(c). Then the (i,j)th element of D(c), the second derivative of $[\sum_{j=1}^{m} d_j(c_j)-D_{max}]^2$ with respect to c_i and c_j, is as follows:

$$D_{ij}(c) = 2\left[\frac{d}{dc_i}d_i(c_i)\right]\cdot\left[\frac{d}{dc_j}d_j(c_j)\right] \qquad 4.3.27$$

$$D_{ii}(c) = 2\left[\frac{d}{dc_i}d_i(c_i)\right]^2+2\frac{d^2}{dc_i^2}d_i(c_i)\left[\sum_{q=1}^{m} d_q(c_q)-D_{max}\right] \qquad 4.3.28$$

D can be presented by a diagonal matrix $\nabla d \nabla d'$, where ∇d is the gradient of d, with ith element $d_i'(c_i)$. The Hessian matrix F(c) of equation (4.3.21) becomes:

$$F(c) = G(c) + D(c)$$

$$= G(c) + 2\beta_h \nabla d \nabla d' + 2\beta_h H(c)\left[\sum_{j=1}^{m} d_j(c_j)-D_{max}\right] \qquad 4.3.29$$

where the diagonal matrix H(c) has ith element $d_i''(c_i)$, the second derivative of $d_i(c_i)$ with respect to c_i. Now the positive definite property of F(c) may not be valid. The choice of β_h must be made very carefully. In the begining, β_h is chosen very small so that $G_i - 2\beta_h\left[\sum_{j=1}^{m} d_j(c_j)-D_{max}\right]H_i$ is always positive. Then the β_h can be larger but

always keep $\beta_{lh}[\sum d_i(c_i)-D_{max}]$ very small so that the Hessian matrix is positive definite around $c^{(n)}$, the nth iterative value.

4.4. Frame Skew Assignment Problem

In this section the problem of the skew assignment of SENET network is studied in order to minimize the data packet delay through the network.

4.4.1. Formulation

The capacity assignment problem has the very good property that both the objective and the constraints are separable functions(i.e. expressed by summations of terms, each term representing the constitution of an individual link). Unfortunately, the frame skew assignment problems do not have such good properties. Another problem is that the skew time is a mod function which is neither linear nor concave. To solve the FA problem is very difficult, so only the deterministic model is tried (i.e. the deterministic assumption in section (4.2), in which the service times of packets are not random variables).

The problem is

$$\text{given : } p_i,\ n_{ij},\ \lambda_{ij}$$

$$\underset{\text{over } s}{\text{minimize}} \sum_{ij} \lambda_{ij}\{[s_j-(s_i+p_i+n_{ij})]\bmod F\} \qquad 4.4.1$$

$$\text{subject to : } 0 \le s_i < F \qquad \text{for all } i$$

To simplify the terms without loss of generality, we will use n_{ij} to present the terms (p_i+n_{ij}) mod F. The propagation delay p_i on link i is absorbed in n_{ij}, the processing time for transfer traffic from link i to link j. The problem becomes

$$\underset{\text{over } s}{\text{minimize}} \sum_{ij} \lambda_{ij}\{[s_j - (s_i+n_{ij})]\bmod F\} \qquad 4.4.2$$

The non-linear mod function is replaced by the following form.

$$z_{ij} = s_j - (s_i+n_{ij})\bmod F = s_j - (s_i+n_{ij}) + k_{ij}\cdot F$$

where k_{ij} is an integer. If we define z_{ij},s_i, and n_{ij} as being in the range [0,F) then k_{ij} can only be one of three values 0, 1, 2. The above non-linear non-concave optimization problem can be formulized as a mixed integer linear program as follows:

$$\text{minimize over } s \quad \sum_{ij} \lambda_{ij} z_{ij} \qquad 4.4.3$$

subject to $0 \le s_i < F$ for all i

$0 \le z_{ij} < F$ for all i,j

$z_{ij} = s_j - (s_i + n_{ij}) + k_{ij}F$

$k_{ij} = 0, 1, 2.$

The above formulation can be simplified. Only the $z_{ij} > 0$ constraints of z_{ij} are effective. Therefore the slack variables z_{ij} can be eliminated and the problem becomes:

$$\text{minimize over } s \quad \sum_{ij} \lambda_{ij}[s_j - (s_i + n_{ij}) + k_{ij}F] \qquad 4.4.4$$

subject to. $0 \le s_i < F$ for all i

$s_j - (s_i + n_{ij}) + k_{ij}F \ge 0$

$k_{ij} = 0,1,2$ forl all i,j.

It seems that the constraint that k_{ij} be a non-negative integer no greater than 2 is also unnecessary. However in the state of the art, the algorithms to solve integer programming are not very well-developed and the upper limit constraints of k_{ij} do prevent the algorithm from drifting away.

4.4.2. Branch and Bound Algorithm

Branch and bound algorithm, an optimization technique that uses the basic tree enumeration, is used to solve the above MILP(mixed integer linear program) [Gar72, Lan73]. A binary tree is formed with each edge imposing a constraint and each vertex

j representing the set of constraines given by the edges along the unique path P_j from v_0 to v_j.

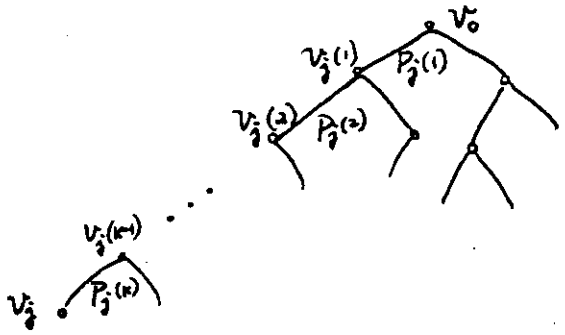


FIGURE 4.4.1 BRANCH AND BOUND TREE ENNUMERATION

If no further exploration from a vertex can be profitable, it is said to be fathomed. More generally, if the problem is to find every $x \in S$, then vertex j restricts x to S_j, where S_j is the intersection with the set of points satisfying the constraints given by the edge P_j. If P_j has k+1 vertices denoted by

$$v_0 = v_{j(0)}, v_{j(1)}, \cdots\cdots v_{j(k-1)}, v_{j(k)} = v_j \qquad 4.4.5$$

then

$$S = S_{j(0)} \supset S_{j(1)} \supset \supset \supset \cdots\cdots \supset S_{j(k)} = S_j \qquad 4.4.6$$

We call $v_{j(k-1)}$ the predecessor of v_j, which in turn is called a successor of its predecessor. If a vertex j does not have any more integer constraints, it is called a tail vertex, or just a tail, i.e. all integer variables of the original problem are fixed by the set of constraints P_j. If the original problem is an all integer program, then the value of the object function at the tails can be obtained by substituting all the constraints. In the MILP cases, the tails, with all the integer variables fixed, have to solve a pure linear program to get the object function.

A vertex that is not fathomed and whose corresponding constraint set can be separated more is called a live vertex. Branching means choosing a live vertex to consider next. The rule of branching considered here is to choose one of the

successor vertices of the vertex currently being considered. If the current vertex j is fathomed, one simply backtracks along P_j until a vertex having at least one live successor is found. One of those successor vertices is chosen for branching. If there are no live vertices, the enumeration is complete.

In a pure enumeration method, every tail is calculated. In branch and bound algorithm, every vertex is estimated by an upper bound and the algorithm will go to its successors only if it is not fathomed, i.e. if there is any possible improvement of the object value. Let the problem be:

$$\max z(x), \qquad x \in S \tag{4.4.7}$$

Suppose the enumeration is at vertex j in the tree. The problem considered at v_j is

$$\max z(x), \qquad x \in S_j \tag{4.4.8}$$

Let

$$z_j^* = \begin{cases} z(x^*(j)) & \text{if } x_j^* \text{ solves equation (4.4.8).} \\ -\infty & \text{if } S_j \text{ is an empty set.} \\ \infty & \text{if equation (4.4.8) is unbounded.} \end{cases} \tag{4.4.9}$$

An upper bound $\bar{z}_j \geq z_j^*$ may be calculated by considering the relaxation of equation (4.4.8).

$$\max z(x) \quad x \in T_j \supseteq S_j \tag{4.4.10}$$

and letting

$$\bar{z}_j = \begin{cases} \infty & \text{if equation (4.4.10) is unbounded} \\ -\infty & \text{if } T_j \text{ is empty, infeasible} \\ z_j^0 = z(x^0(j)) & \text{if } x^0(j) \text{ solves equation (4.4.10).} \end{cases} \tag{4.4.11}$$

The choice of T_j is one of the critical parts of any branch and bound algorithm. It must be chosen such that equation (4.4.10) is relatively easy to solve, but at the

same time must yield an upper bound at a vertex which is valid for any of its successors, since, if v_k is a successor of v_j then $T_j \supseteq S_j \supseteq S_k$. Here the choice of T_j is made as follows:

$$S_j = \{x,y | A_1{}^j x + A_2 y = b^j, \quad x \geq 0 \text{ integer}, y \geq 0\} \qquad 4.4.12$$

$$T_j = \{x,y | A_1{}^j x + A_2 y = b^j, \quad x,y \geq 0\} \qquad 4.4.13$$

so that $\bar{z}_j$ is calculated by solving the corresponding linear program. The algorithm to solve the MILP is as follow:

Let us define BEST=best solution discovered so far satisfying all the discrete constraints.

Algorithm 4.3

[0] (initial) BEST=$-\infty$.

[1] Set the new vertex as vertex j. Solve the linear program T_j. let FUNC=value of the object function of this linear program.

[2] Check the position of vertx j. If it is a tail then go to [3]; otherwise go to [4].

[3] (It is a tail vertex) If the value of the object function FUNC is greater than BEST then substitute the best solution with vertex j, then go to [5].

[4] (It is not a tail vertex) If value of the object function is greater than BEST, this vertex is live, then go to [6]; otherwise this vertex is fathomed, in which case go to [5].

[5] (back up) Current vertex is either a tail or fathomed. Backtracking along P_j until a vertex having at least one live successor is enumerated. Set that live vertex as the new vertex considered. If there are no live vertices, then

stop, the enumeration is complete, output BEST and the corresponding vertex; otherwise go to [1].

[6] (branching) It is probably profitable to go on further. Choose one of the successor vertices of a current vertex as the new vertex considered, go to [1].

4.4.3. Accelerating Algorithm

The above is the standard branch and bound algorithm for MILP. Notice that if a vertex is infeasible, i.e. $\bar{z}_j=-\infty$, it takes solving a linear program to find out. In the problem of equation (4.4.4), there are other intrinsic constraints which make a lot of vertices infeasible. Before we go further, some graph theory terminologies are explained. Intuitivly speaking, a di-graph (directed graph) is a set of points, and a set of arrows, with each arrow joining one point to another. The points are called the nodes of the graph, and the arrows are called the arcs of the graph. An arc is represented by an ordered pair (x,y), where x and y are vertices of the graph, node x is called its initial endpoint, and node y is called its terminal endpoint. Node y is also called a successor of node x, while node x is predecessor of y. If the directions of the arrows in a graph are not specified, the graph is called an undirected graph or just a graph. The arc without any specification of its direction in the undirected graph is called an edge. An undirected graph is callled a simple graph if:

(i) It has no edge of form (x,x).

(ii) No more than one edge joins any two nodes.

A subgraph S of a graph G is a graph with the set of the nodes a subset of nodes of G and the set of arcs a subset of the arcs of G. A chain is a sequence

$q=(u_1,u_2, \cdot, \cdot, u_r)$ of arcs of G such that each arc in the sequence has one endpoint in common with its predecessor in the sequence and its other endpoint in common with its successor in the sequence. A cycle is a chain such that the two endpoints of the chain are the same node. A graph is connected if there exists a chain q(x,y) for each pair (x,y) of distinct nodes. The graph is called separated if it is not connected. A tree is defined to be a connected graph without cycles. Edges in a tree is called branches.

Now look back at the constraints:

$$s_j-(s_i+n_{ij})+k_{ij}F\geq 0 \qquad \text{for all } \lambda_{ij}>0 \qquad 4.4.14$$

$$0\leq s_i$$

For convenience, a delay graph is constructed to show the traffic and the delays of the real communication network. Let the node of the delay graph be the communication links, an arc (i,j) on the delay graph exists only if the traffic λ_{ij} of the communication network is non-zero. Figure 4.4.2 shows an example. Then for the summation of equation (4.4.14) over a cycle L of the delay graph, we will get:

$$\sum_{(ij)\in L} n_{ij} \leq \sum_{(ij)\in L} k_{ij}\cdot F \qquad \text{for every cycle L} \qquad 4.4.15$$

We will prove that equation (4.4.15) is the necessary and sufficient condition for equation (4.4.14) to have a feasible solution. Suppose a system of linear inequalities

$$f_i(x)\geq \alpha_i \qquad (1\leq i\leq p) \qquad 4.4.16$$

where $f_1, f_2, \cdot, ., f_p$ are given linear functionals on x, and $\alpha_1, \alpha_2, ., ., ., \alpha_p$ are given real numbers. The system is said to be consistent, if there exists an x which satisfies the above system; otherwise it is said to be inconsistent.

Theorem 4.1 (Consistence Theorem of [Fan56])

A system $f_i(x)\geq\alpha_i$ for i=1,2, ., .,p, is consistent if and only if every set of positive numbers λ_i satisfies $\sum_{i=1}^{p}\lambda_i f_i=0$ which in turn implies that $\sum_{i=1}^{p}\lambda_i\alpha_i<0$.

Another way to state this theorem is that if a system $f_i(x_i) \geq \alpha_i$, $(1 \leq i \leq p)$ is inconsistent then there exists a set of positive numbers $\{\lambda_i\}$ such that $\sum_{i=1}^{p} \lambda_i f_i = 0$ and $\sum_{i=1}^{p} \lambda_i \alpha_i \geq 0$.

Theorem 4.2

Equation (4.4.14) has a consistent solution for $\{s_i\}$ and $\{k_{ij}\}$ if and only if equation (4.4.15) is satisfied.

Proof : The necessary condition is easy if $\{s_i\}$ and $\{k_{ij}\}$ are consistent then equation (4.4.15) is just the positive linear combination of equation (4.4.14), so equation (4.4.15) will be consistent.

The sufficient condition can be proven by assuming that there exists a set of $\{k_{ij}\}$ which satisfies equation (4.4.15) by not giving a consistent solution of $\{s_i\}$ for equation (4.4.14). Now equation (4.4.14) is just a function of s_i, and it is inconsistent.

$$s_j - s_i \geq n_{ij} - k_{ij} F \qquad \text{for all } \lambda_{ij} > 0 \qquad 4.4.17$$

Then by theory 4.1 there exists a positive linear combination of equation (4.4.17) which makes the left part equal to zero. But this linear positive combination is over the incident matrix of a graph, so this set forms a cycle. This is a contradiction because we just assumed that equation (4.4.15) is valid for every cycle. This proves the theorem. #

Algorithm 4.4 Check the feasibility of the vertex of MILP.

[0] (initial) BEST=$-\infty$, find all the simple cycles of the digraph.

[1a] Set all the nonfixed k_{ij} equal to their upper bounds, i.e. $k_{ij}=2$. Check all the simple cycles to see if $\{k_{ij}\}$ is feasible for inequality equation (4.4.17).

[1b] If $\{k_{ij}\}$ is feasible then solve the linear program with $\{s_i\}$ and all the nonfixed k_{ij}; otherwise set FUNC=$-\infty$, i.e. declare the vertex to be fathomed.

By the above algorithm, the feasiblity of a simple cycle is always checked first before solving the linear problem.

The k-tree algorithm for a simple cycle of a digraph is used in step [0] of algorithm 4.4. [Ber73] The algorithm is based on k-formulas, which are a linear notation for the specification of digraphs. The notation was introduced by Krider [Kri64]. The trees of k-formula are called k-trees. Let N={1,2,3,...n} be the set of nodes of the digraph. Collect the links for all k∈N from which links originate into a tree with node k the root, and the terminal nodes arranged in ascending order from left to right. This tree is called the atomic tree of node k and is denoted t_k.

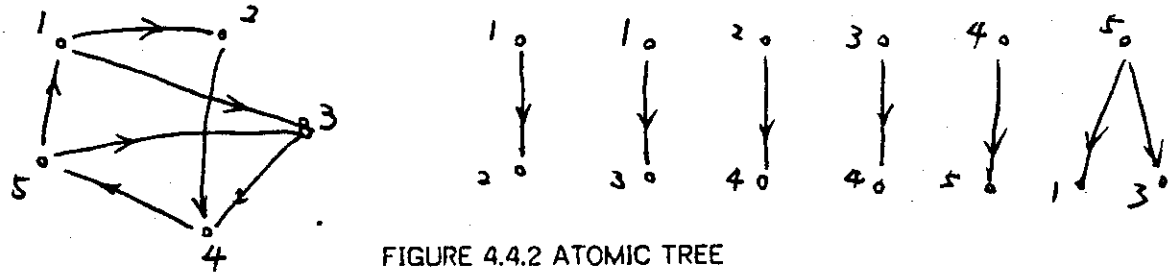


FIGURE 4.4.2 ATOMIC TREE

First the atomic trees of all nodes of the digraph are merged into a single k-tree or a set of k-trees, $T_1, T_2, \ldots, T_m$. In one k-tree, any specific atomic tree can appear at most once. Here is an example of forming of a k-tree.

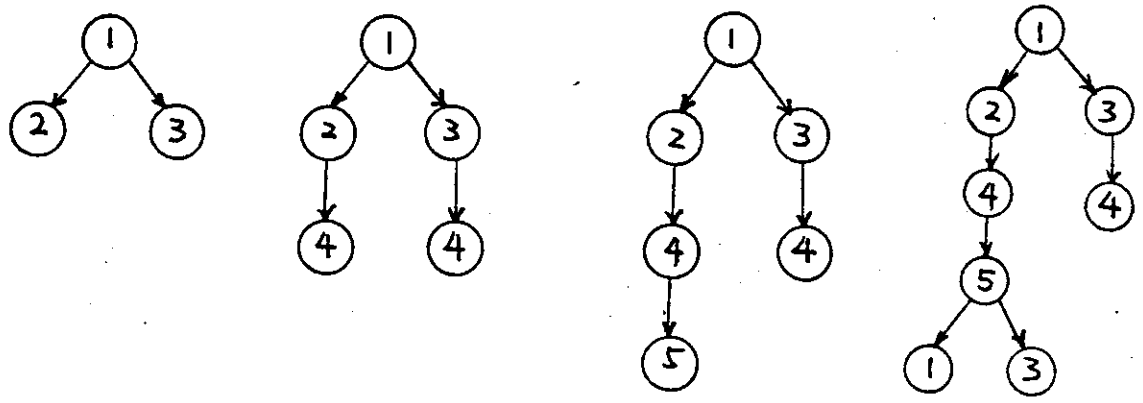


FIGURE 4.4.3 K-TREE

Note that a k-tree must merge as many atomic trees as it can, but for some k-tree it may not be able to merge every atomic tree of the digraph.

Then we can find all cycles of the digraph by traversing the k-trees and checking the path from the root of the tree to the terminal node. If the path contains a subpath $(c_k,\ldots,c_k)$ then one cycle is found. By carefully marking nodes, the duplicates can be eliminated [Ber73].

4.5. Heuristic Algorithm

Even with the accelerating algorithm of section 4.4.3, the mixed integer linear program still runs very slowly. For a communication network with 4 switching nodes and a complete graph topology with duplex links, there are 12 real variables and 24 integer variables. For a pure enumerating method, 3^{24} (each integer has three choices) linear program of 12 variables have to be solved. So inherently the integer programming costs a lot of computation time. Next we will discuss some properties of the optimum solution and develope a heuristic algorithm from that.

4.5.1. Cuts and Trees

Here a graph implies a simple connected graph. A maximum spanning tree T of a connected graph G is a tree subgraph with a set of nodes containing every node of G. Unless otherwise noticed, a tree implies a maximum spanning tree. A cut is a set of arcs in a graph such that if the arcs in the cut are removed from the graph the graph will be separated. The set of fundamental cut relative to a tree T is the set of minimum cut such that each cut contains one link in the cotree G-T of T. Where the minimum cut is a cut such that any strict subset of it is not a cut. A fundamental cut C_i of tree T is a fundamental cut which has branch i of T in it. Arcs in the tree are called branches while arcs in the cotree are called chords.

Lemma 4.3

Suppose a communication network which has links a, b, c with non-zero transfer traffic to link r, i.e. λ_{ar}, λ_{br}, λ_{cr} greater than zero. And suppose t_a, t_b, t_c are the

times when the transfer traffic has finished being processed and they are ready to be sent out. Then the optimum assignment of the frame skew of link r must be equal to either of t_a, t_b, or t_c. Where optimum assignment is defined as the assignment of the least total waiting time.

The general form of the total delay at this junction is the sum of $[(s_r - t_i) \bmod F] * \lambda_{ir}$ where i = a, b, c.

Suppose that the optimum s_r is not equal to any of t_a, t_b, t_c. Without loss of generality, suppose s_r, the skew assignment of link r, lies between t_a and t_b.

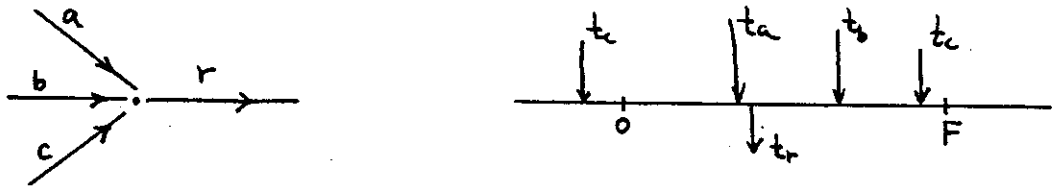


FIGURE 4.5.1 MERGING LINK CONFIGURATION

The delay at the junction will be

$$\begin{aligned}
&(s_r - t_a) \bmod F + (s_r - t_b) \bmod F + (s_r - t_c) \bmod F \\
&= (s_r - t_a)\,\lambda_{ar} + (F - t_b + s_r)\,\lambda_{br} + (F - t_c + s_r)\,\lambda_{cr} \\
&= (\lambda_{br} + \lambda_{cr})F - (t_a\lambda_{ar} + t_b\lambda_{br} + t_c\lambda_{cr}) + (\lambda_{ar} + \lambda_{br} + \lambda_{cr})s_r \\
&= \alpha + (\lambda_{ar} + \lambda_{br} + \lambda_{cr})\cdot s_r
\end{aligned} \qquad 4.5.1$$

It is obvious that decreasing s_r a little will produce a smaller total delay. So it is a contradiction that s_r is the optimum skew assignment. It can be proved similarly for other configuration of skew assignment. So for a single network like this the optimum assignment of link r must be matched to one of the links.

That communication networks with more than three input links merge into one link can be proved similarly, for the third term becomes

$$\sum_{i=1}^{k} (F - t_i + s_r)\lambda_{ir} = (F - t_c + s_r)\lambda_{cr} \qquad 4.5.2$$

with

$$\sum_{i=1}^{k} \lambda_{ir} = \lambda_{cr}$$

and $$t_c = \sum_{i=1}^{k} t_i \lambda_{ir} / \lambda_{cr}. \qquad 4.5.3$$

Next we consider a similar network with a link r which has non-zero transfer traffic to links a, b, c, i.e. $\lambda_{ra}, \lambda_{rb}, \lambda_{rc} > 0$.

FIGURE 4.5.2 SPLIT LINK CONFIGURATION

Where t_a is the skew time of link a minus the necessary processing time for the transfer traffic λ_{ra} and then mod F, or t_a is the perfect matching time of link r to link a. The total delay will be the summation of λ_{ia} times $(t_i - s_r) \bmod F$ over i=a, b, c.

With the same argument, we will have the delay as

$$T = \alpha - (\lambda_{ra} + \lambda_{rb} + \lambda_{rc}) s_r \qquad 4.5.4$$

for some constant parameter α which will remain constant until s_r matches either t_a, t_b, or t_c. It is obvious that increasing s_r a little will always give a better performance if s_r is not equal to any of the times t_a, t_b, t_c. So in this case the optimum of s_r is also matched to one of the links.

Now we will consider a more complicated network.

FIGURE 4.5.3 GENERAL LINK CONFIGURATION

Where t_a, t_b, t_c are the time ready to transfer the traffic λ_{ar}, λ_{br}, λ_{cr},

respectively. And t_d is the skew time of link d minus the necessary processing time for traffic λ_{rd}. t_e, t_f are defined similarly. In other words, those times t_a, t_b, t_c, t_d, t_e, and t_f are the perfect matching times for link r to the specific links.

Lemma 4.4 The optimum skew assignment for link r can always be set to match one of the links.

The proof is very similar to lemma 4.3. Suppose there is an optimum assignment of s_r which is not equal to any of the matching points. By similar argument to that of the proof of Lemma 4.3, the total delay of the network will be of the form

$$T=\alpha+(\lambda_{ar}+\lambda_{br}+\lambda_{cr})s_r-(\lambda_{rd}+\lambda_{re}+\lambda_{rf})s_r$$

If $(\lambda_{ar}+\lambda_{br}+\lambda_{cr})$ is greater than $(\lambda_{rd}+\lambda_{re}+\lambda_{rf})$, then decreasing s_r a little will always produce a smaller delay for the network which is a contradiction of s_r being the optimum assignment. So s_r must match one of the t_i. Otherwise, the increase of s_r will always decrease T by the same argumen. If $(\lambda_{ar}+\lambda_{br}+\lambda_{cr})$ is equal to $(\lambda_{rd}+\lambda_{re}+\lambda_{rf})$ then we can shift s_r without affecting the network performance. So we can always shift s_r until one link is matched. This completes the proof. #

Now we consider a general graph. For any subgraph L of graph N, we will prove that if N is connected, then there is at least one arc which carries non-zero transfer traffic between the communication links in L and the communication links in N-L which is matched, i.e. its transfer delay will be zero. Here the graph is the delay graph which has the original communication links as nodes and the non-zero transfer traffic as arcs between nodes. Each arc on the delay graph has a corresponding traffic and processing time of the communication network.

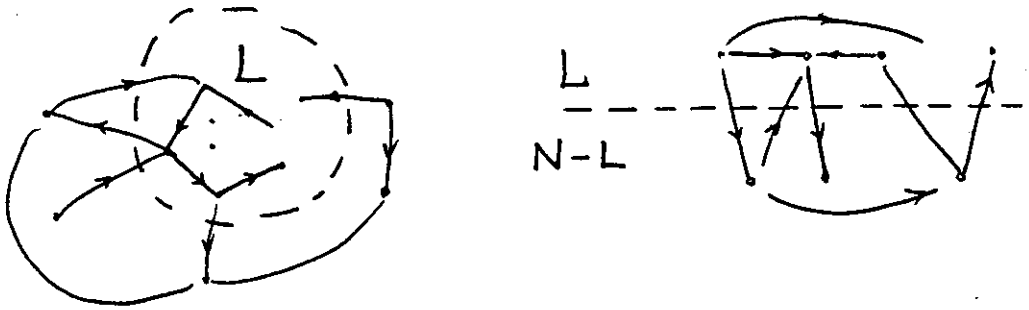


FIGURE 4.5.4 GENERAL NETWORK CONFIGURATION

The argument is very similar to the argument for the above two lemma. If none of the arcs is matched, then the tendency to decrease the skew time will be proportional to all the transfer traffic into L, otherwise, the tendency to increase the skew time is proportional to the transfer traffic leaving L. If the amount of traffic going into L is greater than the amount of traffic leaving L, then decrease the skew time a little for all links of L until one arc is matched. This will always decrease the total delay. The opposite is true when one increase the skew time a little. So in each of these situations the optimum assignment of skew time for L must have at least one matched arc. Therefore the following theory is proven.

Theorem 4.5

For the optimal skew assignment in a delay graph, there exists a connected sub-graph of the matched arcs.

By this theorem, we will only consider the set of skew assignments which has a connected sub-graph of matched arcs. In graph theory, a maximum spanning tree of a connected graph can reach every node, so a maximum spanning tree of matched arcs can determine all the skew of the nodes, or the communication links of the original

network. We will call such a maximum spanning tree as matched tree in skew assignment.

By heuristic, it seems that the tree with maximum total transfer traffic will be the matched tree of the optimum skew assignment. But this is not true. The following example will show the reason and will also explore the complexity of the problem.

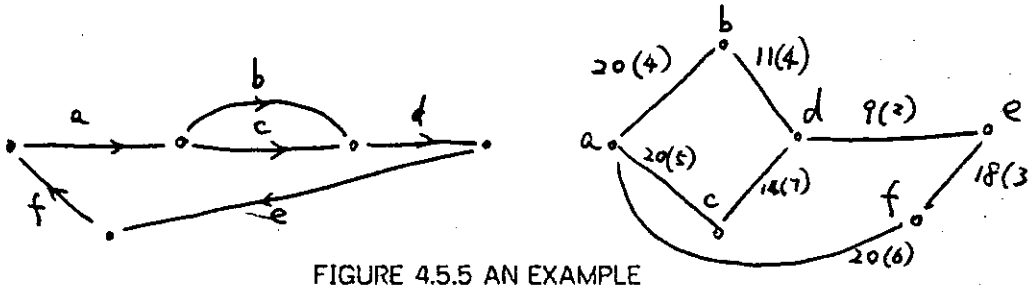


FIGURE 4.5.5 AN EXAMPLE

The number on the arcs of the delay graph is the amount of transfer traffic corresponding to the arc and the number in parenthesis is the necessary processing time for the arc. The frame size is 10.

Then the matched tree of maximum transfer traffic and the corresponding skew assignment is

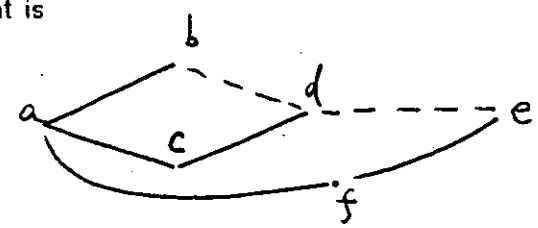


FIGURE 4.5.6 MAXIMUM TRANSFER TRAFFIC SKEW ASSIGNMENT

The total mis-matched delay is

$\lambda_{bd}[12-(4+4)] + \lambda_{de}[11-(2+2)] = 107$

But the matched tree of the optimum assignment is as follows:

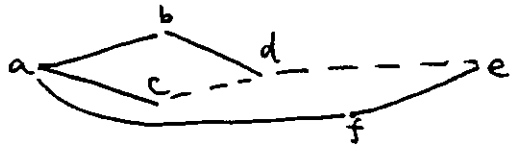


FIGURE 4.5.7 OPTIMUM FRAME SKEW ASSIGNMENT

The total mis-matched delay is

$\lambda_{cd}[18-(5+7)] + \lambda_{de}[11-(8+2)] = 93.$

Why is the first skew assignment not optimum? There are two cycles (abdefa) and (acdefa). The first skew assignment minimizes the cycle (acdefa), and the second skew assignment minimizes the cycle (abdefa). The cycle (abdefa) has total processing time 19, and the cycle (acdefa) has total processing time 23. For a network of frame size 10, the cycle (abdefa) is much more vulnerable than cycle (acdefa). Hence the first skew assignment fails.

To search all the possible trees is infeasible for a moderately complicated network, the number is C^n_{m-1} where m is the number of nodes and n is the number pieces of non-zero transfer traffic. A heuristic algorithm which compares the skew assignments with matched trees differing by only one arc is developed. It is well known in graph theory that the set $\{x_i\} \cup \{T-i\}$ is also a tree for any arc x_i in C_i. Thus we can replace branch i by any arc of C_i and still make a tree. If we choose these trees as matched trees in the skew assignment, then the difference between these skew assignments will only occur at the arcs in C_i. Therefore the algorithm must choose the arc in a fundamental cut such that the mis-matched delay in the cut will be minimum.

4.5.2. The Algorithm

Now consider a matched tree T and its relative skew assignment and a fundamental cut C_i of T. First we mark the nodes on one side of the cut and assume the skew assignment of these nodes to remain constant, because assigning of the skew depends on the direction of the branch. Group all arcs in cut C_i into two sets A and B.

If an arc in the cut is from a marked node to an unmarked node then this arc is in A, otherwise the arc is in B. If we replace branch i by an arc x in the cut as a matched arc. Depending on the direction of the arc x, the change of skew in the unmarked nodes will differ. Define x_r the original mis-matched delay of arc x. If x is in B then every skew assignment of the unmarked nodes should increase an amount x_r ; otherwise x is in A and the skew should decrease x_r for all the unmarked nodes. For an arc y in C_i that is neither i nor x, there will be some change of mis-matched delay during the change of the matched tree. Now suppose the new skew increases by x_r for all the unmarked nodes, then if y is in B the mis-matched delay should subtract x_r amount, otherwise the new mis-matched delay should add x_r amount. The mod F should apply to the arithematical operation after the results are found. A decrease of skew by x_r is equivalent to a increase of skew by $F-x_r$.

Algorithm 4.4

[0] (initial) Read the frame size, the structure of the delay graph that a two column vector LINE of L elements is set with arc I from node LINE[1,I] to node LINE[2,I]. Set the incident matrix which the ith row has 1 only at node LINE[1,I] and LINE[2,I]. Set the processing time of arc j at vector DELAY[J].

[1] Find a tree and set TREE[J]=1 if arc J is in the tree; otherwise TREE[J]=0.

[2] (set the skew assignment of the node by the tree. Mark a node if the skew is assigned.) Set SKEW[1]=0, MARK[1]=1; If MARK[I]=1 and MARK[J]=0 and we will mark node J if there is a arc K between I and J.

SKEW[J]=(SKEW[I]+DELAY[K]) mod FRAME if arc K is from I to J and

SKEW[J]=(SKEW[I]-DELAY[K]) mod FRAME if arc K is from J to I. Set MARK[J]=1;

[3](find the fundamental cut) Inversing the (M-1)X(M-1) square submatrix of the tree and labeling the corresponding column of INCD, the incident matrix will become the fundamental cut matrix. For each row i, the set of the column j such that INCD[I,J]=1 is a fundamental cut with respect to the column which is labeled i. Because one row of INCD is redundant, so INCD[M,K] is used as labeling for all K. Set J=1, and continue [4].

[4](find a tree branch) While INCD[M,J]=0 increment J by 1, if J>L then the algorithm complete, stop; otherwise we have found a tree branch J, Set MINN=J, cntinue [5].

[5]Put the cut corresponding to branch J into vector COTREE, and the number of arcs in the cut is NB; if NB=0, then J=J+1 go to [4]; if NB=1, then if LEM[COTREE[1]]<LEM[J] and arc COTREE[1] is not matched, then there will be profit to switch to the new arc. Set MINN=COTREE[1], go to [11]; if NB>1 then continue [6].

[6]There is a cut corresponding to the branch J of the tree T, mark all the node in one side of the cut.

[7]Set MIN=mis-matched delay with J as branch. Set R=1.

[8]Calculate the total mis-matched delay of the cut with COTREE[R] as the new matched branch, and set the value as NEWMIN.

[9]If NEWMIN<MIN, then set MIN=NEWMIN and MINN=COTREE[R].

[10]Set R=R+1. If R<NB, then we have more arcs to look at, go to [8]; otherwise all the arcs in the cut have been tried, continue [11].

[11]Check if there is a better choice than J. If MINN=J, then J is the best in this cut go to [4]; otherwise eliminate the column MINN of the matrix INCD, so

that only 1 in the column is at row INCD[M,J] where the branch J is labeled. Set INCD[M,MINN] equal to this row, and unlabel the column J of INCD. The elimination process is the same as in [3] where the set of fundamental cuts is found.

[12]We get a new branch, reassign the unmarked node by increase ing or decreaseing the same amount XR such that it makes branch MINN matched. Set J=1, go to [4].

The above is the main frame of the algorithm. Step [3] is to inverse some of the incident matrix and find the fundamental loop. Every iteration will find a new tree and increase the performance. There is a finite number of trees, so the algorithm will terminate. This algorithm will find a matched tree such that it is better than all its immediate neighborhoods where one arc is different. Thus different initial guesses of matched trees may result in different sub-optimum configurations of matched trees.

CHAPTER 5 Conclusion

This dissertation describes the model, the analysis and the design of a special integrated voice/data computer communication system. The advantages and the operation constrains of the integrated switch are thoroughly studied. This dissertation gives guidances of the expected performance and the potential storage requirement. This chapter presents interpretations of results and indicates directions for future research.

5.1. Interpretation of Results

In order to meet certain grade of service for voice calls, the communication system must have some redundant facility in addition to its traffic requirement. The idea of the integrated switch SENET communication network is to use these redundant facilities for data packet communication. In Chapter 2, an integrated switch system of ten voice channels and of job size ratio 10 and 100 is solved exactly. The exact solutions are also compared with various approximation methods and with results of similar queueing models. The rationale of the very long data packet queue for the integrated switch is thus concluded by the queue occasionally entering the overloaded states for a relatively long period. From the heavy load approximation of queueing theory, we believe that the diffusion approximation should be a better approximation when the scale of the integrated switch increases and be relatively insensitive to the distribution of the service requirement. Though the quantitative results of a large integrated switch with the realistic job size ratio need further investigation, the

qualitative results do give a good meaningful estimation of the potential problems. The use of the redundant voice communication facilities for data communication will create a long queue of data packets and will have slow response for interactive data traffic. It is thus conceived that to have real time response for interactive data communication, some link capacities shall be reserved for such traffic only. In addition to this, the integrated switch needs a much larger memory space to queue the data packets than the conventional data communication processors such as ARPA IMP.

Although the data packet queue of integrated switch is rather long, the number of buffer needed for the switch to operate efficiently is limited. The queueing network of Section 3.3 models the buffer life-time and the relationship of throughput and number of buffers. The concept of processing bound and buffer bound switching processor is developed. In order to minimize the blocking probability for Class II packets by lacking of buffer and to maximize the memory utilization, a secondary storage is suggested and modelled properly. The use of a disk for switching processor is rather unconventional, a special reservoir model and a special forward and backward algorithm are developed. The block transfer time between disk and core is either Exponential distributed or Erlang distributed. It is shown that the performance of the switching processor with disk is insensitive to the distribution of transfer time. Using a disk for the switch will decrease the buffer requirement and isolate the overloaded switch from infecting its neighboring nodes. However the disk may create some reliability problems and also may complicate the operations of network. Several schemes to allocate memory into buffers are also discussed and their performance are compared with each others. Both first-fit assignment scheme and dynamic assignment of fixed-size buffer scheme sacrifice the larger size packets too much in a heavy

loaded system. However both give superior memory management schemes than other schemes in a lightly loaded system. A reserve priority buffer assignment scheme is suggested for communication network with priority on its data packets. High priority packets can be allocated more buffers than low priority packets can. A small sacrifice on the low priority packets can decrease the blocking probability and thus shorten the response time of high priority packets a lot.

In order to have real time response to the interactive data communication, some communication link capacity shall be reserved for Class II traffic only. The link capacity assignment to interactive data is discussed as a grade of service trade-off between Class I and Class II traffic. The SENET communication network with frame structure and the concave cost-capacity function is solved by the coordinate descent method and Newton's method. The capacity assignment problem for the SENET network is different form other communication networks and the link capacities shall be re-assigned when the traffic situation changes. Another problem related to the frame structure of SENET is the frame skew assignment problem. Because of the frame period, the delay of packets is discretely incremented by the frame period. The mis-matching of the frame skew between links will cause the longer buffer life-time, thus require larger memory space as well as suffer extra mis-match delay. The frame skew assignment problem is formulated as a mixed integer linear program. A special speeding algorithm for the global optimal assignment as well as a heuristic algorithm for local optimal assignment is developed. This frame skew assignment also needs dynamically re-assignment when the traffic flow on links changes.

5.2. Directions for Future Research

With the changing cost structure of processing power and communication power, many integrated switch networks will be designed and used. The nature of voice and data communications are still unclear, and the relationship between them are also left unexplored. The flow control in an integrated switch network is especially important to the network performance, while it is totally unknown yet. Possible further studies are outlined below:

1. With a backbone communication network using T1 carrier, a 10 millisecond frame contains 15,440 bits. If half of the capacity can be used by Class I traffic and each voice slot contains 80 bits, then the integrated switch has 97 Class I channels. For a 2000 bits packet, a T1 carrier can transmit the whole packet in about 1.3 millisecond. Comparing 1.3 millisecond with the average 3 minutes voice call, we find that the realistic Class I/Class II job size ratio is in the order of 100,000. An integrated switch of such order can be solved by diffusion approximation, but the validation of such approximation either by simulation or other methods still needs lots of efforts.

2. The control protocol for the integrated switch communication network especially with disks needs a lot of work to find the impact of the present protocol and the effectiveness of the disk. The feasibility of using CCD or magnetic bubble as a secondary storage device and their corresponding models will lead the communication processor into a better cost-performance region.

3.The processing of data packets is assumed to be Exponential distributed. A simulation for general processing time will give a better understanding of buffer life-time behavior and might suggest a better buffer management scheme.

4.The cost-capacity function discussed in Chapter 4 is assumed to be link independent. However, there are some relationship between the cost of adjacent links especially in SENET the cost function is the grade of service of Class I traffic. A study of this will give a more clear picture of data flow in the network and can provide design guidance of flow control and network protocol.

BIBLIOGRAPHY

[ADC75] *Advanced Data Communication Control Procedures (ADCCP), Independent Numbering.* Proposed American National Standard. Fourth Draft, August 1975.

[And72] Anderson, R. R., J. .F. Hayes, and D. N. Sherman "Simulated Performance of a Ring-Switched Data Network." IEEE Trans. on Communications, 1972.

[Avi61] Avi-Itzhak, B. and P. Naor "On a problem of Preemptive Priority Queueing." Operations Research Vol. 9, 1961.

[Bar76] Barbacci, M. R. and J. D. Oakley "The Integration of Circuit and Packet Switching Networks Toward a SENNET Implementation." The 15th NBS-ACM Annual Technique Symposium, 1976.

[Bas75] Baskett, F. et al. "Open, Close, and Mixed Networks of Queues with Different Class of Customers." JACM Vol. 22, 1975

[Ber73] Bertiss, A. T. "A K-tree Algorithm for Simple Cycles of a Directed Graph." Technical Report 73-6 Dept. of Computer Science, University of Pittsburgh. 1973.

[Bha75] Bhat, U. N. and M. J. Fischer " Multichannel Queueing System with Heterogeneous Classes of Arrivals." Defence Communication Engineering Center CP-75010, 1975.

[Bro68] Brosh, I. "Preemptive Priority Assignment in Multichannel System." Operations Research Vol. 16, 1968.

[Buz73] Buzen, P. " Computational Algorithm for Closed Queueing Network with Exponential Servers." CACM Vol. 16, 1973.

[Can72] Cantor, D. and M. Gerla "The Optimal Routing of Messages in a Computer Network via Mathematical Programming." IEEE Computer Conference Proceedings, pp. 167-170, San Francisco, Calif., September 1972.

[CMU75] Carnegie-Mellon University "The Application of Multiple Processor Computer Systems to Digital Communications Networks." CMU Proposal No. 08103A to Defense Communications Engineering Center, Defense Communications Agency, May 1975.

[Cha43] Chandrasekhar, S. "Stochastic problem in Physics and Astronomy." Rev. Mod. Phy. 15, 1943.

[Cha72B] Chandy, K.M. and R.A. Russel "The Design of Multipoint Linkages in a Teleprocessing Tree Network." IEEE Trans. on Computers, C-21, pp. 1060-1066, 1972.

[Cha75A] Chandy, K. M., U. Herzog, and L. Woo " Parametric Analysis of Queueing Networks." IBM. J. Res. Develop. 1975.

[Cha74] Chang, A. and S. S. Lavenberg "Work Rates in Closed Queueing Networks with General Independent Server." Operations Research, Vol. 22, pp. 838, 1974.

[Cha75B] Chandy, K. M., U. Herzog, and L. Woo " Approximation Analysis of General Queueing Network." IBM. J. Res. Develop. 1975.

[Cha72a] Chang, J. H. " Some Design and Evaluation Techniques of Data Communication Systems." IBM. Research Report RC-3736, 1972.

[Chu69] Chu, W. W. "A Study of Asynchronous Time-Division Multiplexing for Time-Sharing Computer Systems." FJCC. Vol.35, pp. 669, 1969.

[Chu72] Chu, W. W. and A. G. Konhein " In the Analysis and Modeling of a Class of Computer Communication Systems." IBM. Res. Report RC-3727, 1972.

[Clo73] Closs, F. "Pakcet Arrival and Buffer Statistics in a Packet Switching Node." Data Network Analysis and Design, DATACOMM 73, 3rd data communication Symposium.

[Cob54] Cobham, A. "Priority Assignemtn in Waiting Line Problem." Operations Research Vol. 2, 1954.

[Col71] Cole, G. D. "Performance Measurements on ARPA Computer Network." Proceedings of Second ACM IEEE Symposium on Problems in the Optimization of Data Communication Systems, Palo Alto Calif. October 1971.

[Cov75] Coviello G. J. and P. A. Vena "Integration of Circuit/Packet Switching in a SENET (Slotted Envelope NETwork) Concept." National Telecommunications Conference (NTC), New Orleans, December 1975.

[Cox55] Cox, D. R. "A use of Complex Probabilities in the Theory of Stochastic Process." Proc. Cambridge Phil. Soc. 51, 1955.

[Doi69] Doll, D. R. "Efficient Allocation of Resource in Centralized Computer-Communication Network Design." Univ. of Michigan, AD 862 281, 1969.

[Doi71] Doll, D. R. "Topology and Transmission Rate Consideration in the Design of Centralized Computer Communication Network." IEEE Trans. on Communications, COM-19, 1971.

[Fel66] Feller, W. *An Introduction to Probability Theory and Its Applications Vol.II.* John Wiley, New York, 1966.

[Fis75] Fischer, M. J. and T. C. Harris " An Analysis of an Integrated Circuit and

Packet Switched Telecommunication System." Defense Communication Engineering Center, TN 6-75, 1975.

[For75] Forgie, J. W. "Speech Transmission in Packet Switched Store-and-Forward Networks." Proceedings of the National Computer Conference, NCC75, pp. 137-142.

[Fra71] Frank, H., I.T. Frisch, W. Chou, and R. Van Slyke, "Optimal Design of Centralized Computer Networks." Networks, Vol. 1, pp. 43-57, 1971.

[Fra72] Frank, H., R. E. Kahn, and L. Kleinrock "Computer-Communication Network Design-Experience with Theory and Practice." AFIPS Conference Proceedings, Vol. 40, pp. 255-270, SJCC, Atlantic City, N.J., 1972.

[Gar72] Garfindel, R. S. and G. L. Nemhauser *Integer Programming*. New York, Wiley 1972.

[Gau67] Gautschi, W. "Computational Aspects of three-term Recurrence Relations." SIAM Review Vol. 9, No. 1, Jan. 1967.

[Gav66] Gaver, D. "Observing Stochastic Processes and Approximation Transform Inversion." Operations Research Vol. 14, 1966.

[Gav68] Gaver, D. P. "Diffusion Approximations and Models for Certain Congestion Problems." Journal of Applied Probability, Vol. 5, pp. 607-623, 1968.

[Ger73] Gerla, M. "The Design of Store-and-Forward (S/F) Networks for Computer Communications." U.C.L.A. Report, AD 758 704., 1973.

[Gor67] Gordon, W. J. and G .F. Newell "Closed Queueing System with Exponential Servers." Operations Research Vol. 15, pp. 254, 1967.

[Gra71] Graham, R. J. and H. O. Pollak "On the Addressing Problem for Looping Switching." Bell System Technical Journal, Vol. 50, 1971.

[Had64] Hadley, G. *Non-linear and Dynamic Programming*. Addison-Wesley, Reading, Mass., 1964.

[Hay71] Hayes, J. E. and D. N. Sherman "Traffic Analysis of a Ring Switched Data Transmission System." Bell System Technical Journal Nov. 1971.

[Hea70] Heart, F. E., R. E. Kahn, S. M. Ornstein, W. R. Crowther, and D. C. Walden "The Interface Message Process for the ARPA Computer Network." AFIPS Conference Proceedings, Vol. 36, pp. 551-567, SJCC, Atlantic City, N.J., 1970.

[Hea73] Heart, F. E., S. M. Ornstein, W. R. Crowther, and W. B. Barker "A New Minicomputer/Multiprocessor for the ARPA Network." Proceedings. AFIPS NCC 42, 1973, pp. 529-537.

[Her75] Herzog, U., L. Woo and K. M. Chandy "Solution of Queueing Problems by a Recursive Technique." IBM. J. Res. Develop. 1975.

[Hou70] Hough, R. W. "Future Data Traffic Volume." IEEE., COMPUTER, Vol. 3, No. 5, September/October 1970, pp. 6-12.

[Igl65] Iglehart, D. "Limiting diffusion approximation for the many server queue and the repairman problem." J. Appl. Prob. 2, 1961.

[Ito73] Itoh, K. and T. Kato "An analysis of Traffic Handling Capacity of Packet Switched and Circuit Switched Networks." DATACOMM 73, 1973.

[Jac63] Jackson, J. R. " Jobshop-like Queueing System." Management Sci. Vol. 10, 1963.

[Jon74] Jones, W. B. and W. J. Thron " Numerical Stability in Evaluating Continued Fractions." Mathematics of Computation Vol. 28, 1974.

[Kim75] Kimbleton, S. R. and G. M. Schneider "Computer Communication Networks Approaches, Objectives, and Performance Considerations." ACM Computing Surveys, Vol. 7, No. 3, September 1975, pp. 129-173.

[Kin64] Kingman, J. F. C. "The heavy traffic approximation in the theory of queues." Chapter 6 in Proceedings of the Symposium on Congestion Theory. Editor W. L. Smith and W. E. Wilkinson. No. 2 in Univ. of North Carolina Monograph Series in Probability and Statistics. 1964.

[Kle72] Kleinrock, L. *Communication Nets Stochastic Message Flow and Delay.* Dover Publication, Inc. New York, 1972.

[Kle74A] Kleinrock, L. "Resource Allocation in Computer Systems and Computer-Communication Network." Information Processing 74, 1974.

[Kle74B] Kleinrock, L. "On Measured Behavior of the ARPA Network." AFIPS Conference Proceedings, Vol. 43, pp. 764, NCC, 1974.

[Kle75] Kleinrock, L. and F. A. Jobagin "Packet Switching in Radio Channels Part I and Part II." IEEE Trans. on Communications, COM-23, pp. 1400-1417, 1975.

[Kob74a] Kobayashi, H. "Application of the Diffusion Approximation to Queueing Network I Equilibrium Queue Distributions." Journal of ACM Vol. 21, No. 2, 1974.

[Kob74b] Kobayashi, H. "Application of the Diffusion to Queueing Networks II Nonequilibrium distributions and Applications to Computer Modeling." Journal of ACM, Vol. 21, No. 2, July 1974.

[Kuc72] Kuczura, A. " Queues with Mixed Renewal and Poisson Input." Bell System Tech. Journal, Vol. 51, 1972.

[Kum74] Kummerle, K. "Multiplexor Performance for Integrated line and Packet-Switch Traffic." The Second International Coference on Computer Communication 1974.

[Lan73] Land, A. H. and S. Powell *FORTRAN Codes for Mathematical Programming Linear, Quadratic and Discrete.* London, New York, Wiley. 1973.

[Lue73] Luenberger, D. G. *Introduction to Linear and Nonlinear Programming.* Reading, Mass., Addison-Wesley Pub. Co. 1973.

[Lyo74] Lyons R. E. "Computer Communications in the Department of Defense." 13th Annual Technical Symposium, ACM Washington D.C. Chapter, Gaithersburg, Md. June 1974.

[Mar72] Martin, J. T. *System Analysis for Data Transmission.* Englewood Cliffs, N.J. Prentice-Hall, 1972.

[Mei71] Meister, B., H. Muller and H. Rudin "Optimization of a New Model for message-switch Network." Proc. of Internation Conference on Communication 1971.

[Mit68] Mitrany, I. and Avi-Itzhak "A Many-Server Queue with Service Interruptions." Operations Res. Vol. 16, pp. 628, 1968.

[New75] Newell, A. and G. Robertson "Some Issues in Programming Multi-mini Processors." Department of Computer Science, Carnegie-Mellon University, June 1975.

[Pie72] Pierce, J. R. "Network for Block Switching of Data." Bell System Technical Journal, Vol. 51, 1972.

[Ped72] Pederson, R. D. and J. C. Shah "Multiserver Queue Storage Requirement with Unpacked Messages." IEEE Com-20, 1972.

[Pos68] Posner, M. and B. Bernholtz "Closed Finite Queueing Networks with Time Lags." Operations Research Vol. 16, 1968.

[Red76] Reddy, D. R. "Speech Recognition by Machine A Review." IEEE Proceedings, April 1976.

[Rob70] Roberts, L. G. and B. D. Wessler "Computer Network Development to Achieve Resource Sharing." SJCC Proceedings, Vol. 36, 1970, pp. 543-549.

[Ros75] Rosner, R. D., R. H. Bittel and D. E. Brown "A High Throughput Packet-Switched Network Techniques without Message Reassembly." IEEE COM-23, 1975.

[Sch67] Schrage, L. E. "The Queue M/G/1 with Feedback to lower priority Queues." Management Sci. Vol. 13, 1967.

[Seg69] Segall, M. "A Multi-server System with Preemptive Priorities." Operatons Research Vol. 17, 1969.

[Sie75] Siewiorek, D. P. and M. R. Barbacci "Modularity and Multiprocessor Structures Some Open Problems in the Construction and Utilization of Mini- and Micro-Processor Networks." To appear in the Infotech State of the Art Report on Distributed Systems.

[Ver74] Verma, P. K. and A. M. Rybczynski "The Economics of Segregated and Integrated Systems in Data Communication with Geometrically Distributed Message length." IEEE COM-22, 1974.

[Wal74] Wald, L. D. "Integrated voice/data compression and multiplexing using associative processing." AFIPS Computer Proceedings, Vol. 44, pp. 133, 1974.

[Whi70] Whitney, V. K. M. "A Study of Optimal File Assignment and Communication Network Configuration in Remote-Access Computer Message Processing and Communications Systems." University of Michigan SEL Technical Report No.48, September 1970.

[Wol65] Wolman, E. "A Fixed Optimum Cell-Size for Records of Various Length." Journal of ACM, Vol. 12, No. 1, 1965.

[Wul72] Wulf, W. A. and C. G. Bell "C.mmp A Multi-Mini-Processor." Proceedings of the FJCC, 1972.

[Wul75] Wulf, W. A., R. Levin, and C. Pierson "An Overview of the HYDRA Operating System Development." Proceedings of the 5th ACM Symposium on Operating Systems Principles. Austin, Texas, November 1975.

[Yag71] Yaged, B. Jr. "Minimum Cost Routing for STatic Network Models." Networks, 1, 1971.

[Zaf74] Zafiropulo "Flexible Multiplexing for Networks Supporting Line-Switched and Packet-Switched Data Traffic." ICCC 1974 Conference, Stockholm, Sweden.

[Zei71] Zeigler, J. F. "Nodal Blocking in Large Networks." School of Engineering and Applied Science, Univ. Of California, Los Angles, UCLA-ENG-7167, 1971.