August, 1976

Report ESL-R-678

LOAD SHARING IN A COMPUTER-COMMUNICATION NETWORK

BY

Eberhard Frank Wunderlich

This report is an extension of a thesis supervised by Professor John M. Wozencraft submitted to the Department of Electrical Engineering and Computer Science in partial fulfillment of the requirements for the degree of Master of Science, September, 1975. This work was supported by a Sloan Research Traineeship, a Vinton Hayes Fellowship and the Advanced Research Projects Agency under Contract N00014-75-C-1183.

Electronic Systems Laboratory
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

ABSTRACT

This study investigates load sharing in a system of computers interconnected by a store and forward communication network. The problem is analyzed by modeling both computers and communication channels as queues and evaluating system performance on the basis of the steady state expected time to process computer jobs in the system. Upper and lower bounds on this performance criteria are developed and used to define regions of operation for a network using load sharing. Two techniques for load sharing are then presented.

The first technique, called statistical load sharing, consists of sending a fraction of the jobs arriving at overloaded computers to underloaded computers by random sampling. This technique is analyzed by a network of queues model. It is shown that the general formulation of statistical load sharing is a nonlinear multicommodity flow problem which can be solved by an efficient computer algorithm. The improvement in system reliability due to the ability of load sharing to provide emergency backup in case of computer failure is also studied.

The second technique for load sharing, a type of dynamic load sharing, makes job assignments to computers on the basis of the computers not busy at the time of assignment. This technique is analyzed by an approximation to the hypercube queueing model.

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation for the assistance of my thesis supervisor, Professor John M. Wozencraft. Numerous discussions with him contributed greatly to the completion of this work.

I would also like to acknowledge the contribution of Joe Defenderfer, who is largely responsible for the general formulation of statistical load sharing as a multicommodity flow problem and who implemented the computer algorithm used to solve this problem.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

CHAPTER I INTRODUCTION

1.1 Description of the Problem

Computer-communication networks are a major area of technological development today. The main reason for the current interest in computer-communication networks is that such networks are able to provide system capabilities that far surpass the capabilities of a single isolated computer. Some of the system capabilities that combined communication and computer systems can provide are:

1) Remote, interactive access to time-shared facilities.
2) Sharing of computer data bases.
3) Sharing of specialized computer resources.
4) Load sharing among computers.
5) Emergency backup in case of computer failure.

Ref. [34]

The purpose of this study is to quantify some of the load sharing and emergency backup benefits that a computer-communication network can provide.

The specific problem studied is load sharing in a system of computers interconnected by a store and forward communication network. The problem is analyzed by modeling both computers and communication channels as queues. This model is of interest because it is mathematically tractable. It allows one to evaluate the performance of a system of computers in

terms of the steady state expected time to process a computer job, which is the performance measure used in this study. The time to process a computer job includes the computation time plus any communication time the job may require if it is processed at a computer other than the one at which it was submitted. This concept of load sharing is illustrated in Figure 1.1.

This study shows that one of the possible benefits of load sharing is a lower expected time to process jobs in the system. A second benefit of load sharing is increased system reliability due to the ability to provide emergency backup. A more detailed summary of these results is given after the following brief discussion of the use of store and forward communication networks to interconnect computers and previous studies of load sharing.

1.2 Background

This study considers store and forward (message switched) communication networks since the queueing model used to represent such networks makes the load sharing problem mathematically tractable. Moreover, much recent progress in the area of resource sharing among computers using packet switched communications, a form of message switching in which long messages are divided and sent as several packets, has been made by the Advanced Research Projects Agency (ARPA) Network. The ARPA Network is a distributed packet switched system which ties together many of the

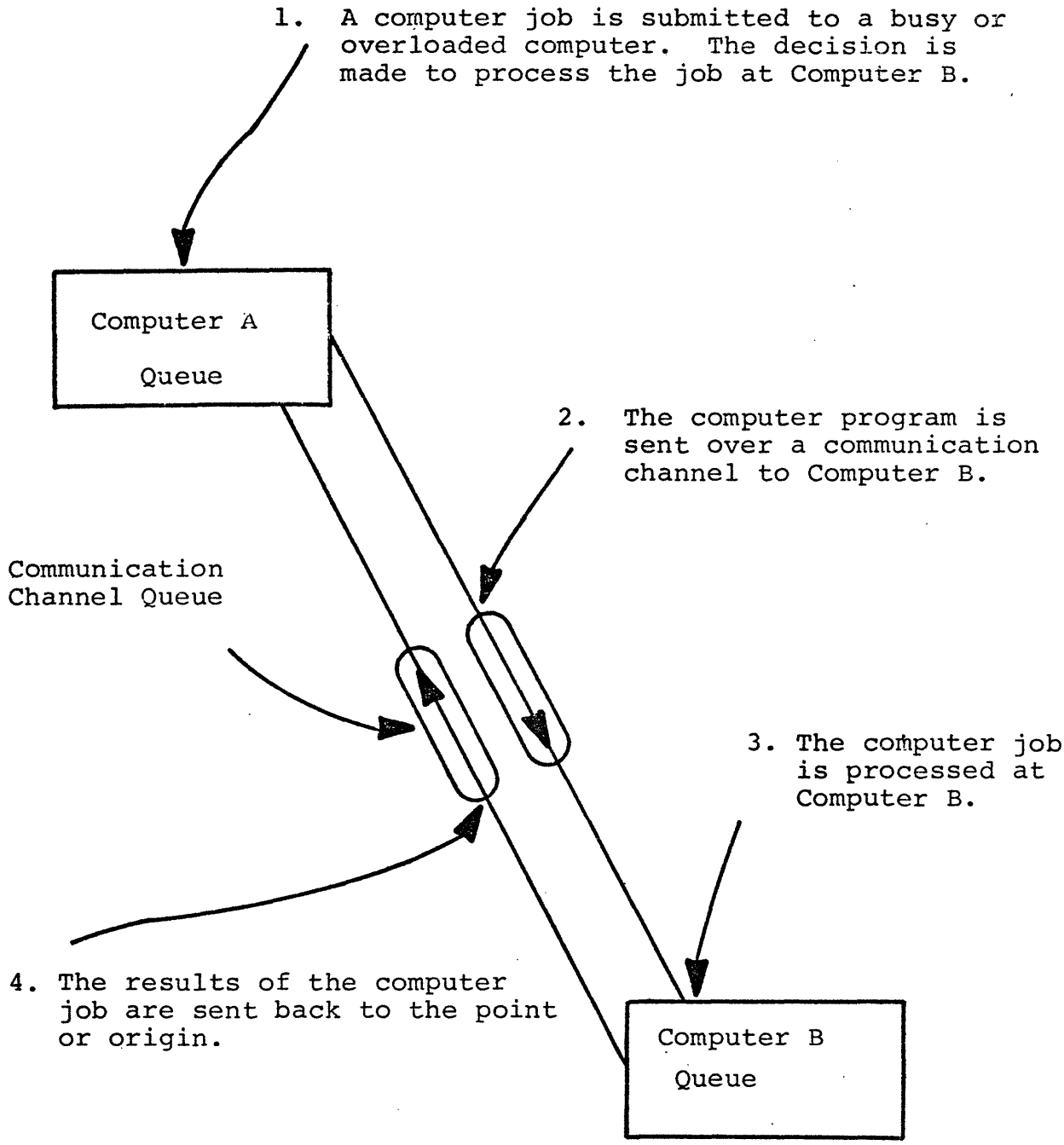


Figure 1.1 The Concept of Load Sharing. This figure shows the sequence of events that occur when load sharing is used in a computer system.

major research computer facilities in the United States. The goal of the network is to make every local computer resource, both hardware and software, available to any user in the network without degradation. In attempting to meet this goal economically, it was found that a distributed packet switched network was an attractive design choice, as has been discussed by Kahn [Ref. 15] and Roberts and Wessler [Ref. 30].

One of the reasons that packet switched communication networks are appropriate for computer communication is that computer traffic tends to be bursty in nature. [Refs. 7 and 13] Packet switching allows one to make good use of communication facilities when the traffic handled is of this type. This is because in a packet switched system, there is no need to switch communication circuits between source and destination before and after each burst of traffic. Instead, the communication circuits in the network can be shared by messages with different destinations without incurring circuit switching delays which tie up the circuits while not allowing data to be transmitted over them. [Ref. 28]

The ARPA Network experience has brought about the serious consideration of packet switched communication networks as the design choice for future computer-communication networks. This supports the study of load sharing using a store and forward communication network.

Load sharing in a network of computers has previously been studied by Bowdon [Refs. 1 and 2]. The study considered a network of computer centers in which jobs of different priority classes were processed. The

computers within each center were modeled as queues with finite length buffers, making the system a system with loss. For this network of computer centers, a load sharing algorithm to improve network throughput was proposed. The dispatching algorithm was to balance the load in the network so that, for each priority class, the expected waiting times at all computers would be equal. A quantitative measure of the improvement in system performance achieved by using the load sharing algorithm was not given.

Roome and Torng [Ref. 31] have studied a type of dynamic load sharing in a computer-communication network where jobs are assigned to computers for processing on the basis of the expected time to process them at the various computers in the system. They have shown by way of simulation that improvements in expected job time in a distributed computer system can be achieved by this technique.

Another study of load sharing in a computer network has recently been done by McGregor and Boorstyn [Ref. 27]. Their study developed a model for load sharing operation in which both computers and communication channels were modeled as queues. Computer jobs were dispatched to various computers by random sampling and a modified gradient algorithm was used to find the load sharing policy which minimized the expected job time in the network. This problem formulation is identical to what is called statistical load sharing in this report. The McGregor and Boorstyn study was done prior to and independently of this report. There is substantial overlap in the two studies and where such overlap occurs, the results agree. This report studies some of the characteristics of statistical

load sharing in greater detail than the previous work and shows how one can apply an existing efficient algorithm for solving multicommodity flow problems directly to the load sharing problem.

McGregor also studied the problem of how to design optimum computer-communication network topologies in which load sharing was to be used [Ref. 26]. Heuristic algorithms were presented for the design of tree topologies which minimize the weighted sum of network cost and expected job time, and the design of connected topologies which maximize throughput subject to a network cost constraint and a maximum expected job time constraint.

1.3 Summary of Results

The queueing models used to represent computers and store and forward communication channels, along with the validity of the modeling assumptions, are discussed in Chapter 2. Upper and lower bounds on system performance in terms of steady state expected job time are then developed as follows. Upper bounds are developed by considering systems with an infinite capacity communication network and an instantaneous global controller. The performance of a system of computers without intercommunication is used as a lower bound on performance. The upper and lower bounds are then used to define two regions of load sharing operation. The first region represents an improvement in expected job time due to the correction of average load imbalances in the system.

Operation in this region can be achieved by a technique that is called statistical load sharing, which is investigated in Chapter 3. The second region of improvement is due to the benefits of using a large system, rather than many small individual systems, when job assignments are made on the basis of the system state at the time of the assignment. Operation in this region is called dynamic load sharing, a limited technique for which is investigated in Chapter 4.

Statistical load sharing improves system performance by sending a fraction of the jobs that arrive at overloaded computers to underloaded computers by random sampling. The analysis of this load sharing technique in Chapter 3 starts by considering a number of specific examples to show some of its main operating characteristics. It is shown that the correction of load imbalances by statistical load sharing can significantly improve the expected job time in the system at high loads. Most importantly, load sharing using an adequate communication network can increase the maximum possible system throughput with a load imbalance in the system. After considering the specific examples, it is shown that the general formulation of the statistical load sharing problem is a nonlinear multicommodity flow problem that can be solved by an efficient optimization algorithm. The algorithm has been implemented [Ref. 6] and examples are given of its use.

The final topic investigated in Chapter 3 is load sharing operation with failure in the system. It is shown that load sharing can increase

system reliability by making the system fail soft, i.e., if one computer in the system fails, the system can continue to operate at reduced capacity. Operation at this reduced capacity, however, can increase the expected job time considerably and this degradation must be accounted for in system design. It is also shown that the failure of a communication link in a load sharing system can increase the expected job time significantly.

Statistical load sharing can improve system performance only by balancing average loads. It is possible to achieve performance gains beyond those attainable by such load sharing by making job assignments to computers dynamically on the basis of which computers are available at the time of assignment† rather than by random sampling. Chapter 4 presents a way of doing this by operating the system using a global controller that assigns jobs on a first-come-first-serve basis. Jobs are assigned to the computer at which they were submitted, if it is not busy, or to the first available computer in the system according to a preference list, if the computer of origin is busy. This load sharing technique is analyzed by an approximation to the hypercube queueing model which represents such operation. It is shown that using a com-

---

†The dynamic load sharing technique studied here differs from that studied by Roome and Torng.

munication network of sufficient capaicty, dynamic load sharing can provide gains in expected job time beyond those achievable by statistical load sharing.

Chapter II MATHEMATICAL MODELS FOR COMPUTER-COMMUNICATION NETWORKS AND BOUNDS ON SYSTEM PERFORMANCE

2.1 Model of a Computer

The model of a computer used in this study is the simplest model of a computer operating in a batch processing mode. This model is the single server queue with a Poisson input stream of jobs and a negative exponential service time distribution (M/M/1 queue) [Ref. 20]. The specific assumptions that are made when using this model are:

1. The input stream of jobs is a Poisson process with mean arrival rate λ.

2. The number of operations required per job is distributed as a negative exponential with mean $1/\ell$.

3. The computer performs R operations per unit time. This means that the service time per job (not including waiting time) is distributed as a negative exponential with mean $1/\ell R$.

4. Jobs are processed in a first-come-first served manner. If the computer is busy when a job arrives, it is queued in an infinite buffer.

The validity of these assumptions depends of course on the specific system being studied, there being a wide range of computing systems in use today. For example, the inputs to the computer could be batch programs read in through a card reader, inputs from an interactive terminal or inputs from a remote sensor. The assumption of a Poisson input stream may or may not hold for the system under consideration. For the case of inputs from a teletypewriter-like terminal, Fuchs and Jackson

[Ref. 7] have shown that the interarrival time between user inputs often fits a gamma distribution which can sometimes be approximated by an exponential distribution as required for the input stream to be Poisson.

The assumption of an exponential service time distribution is an important one to examine. It is important because the performance evaluations made in this study are based on an expected job time criteria, and the service time distribution of a queue has a direct influence on this parameter. The well known Pollaczek-Khintchine formula gives the expected number of customers in a single server queueing system with Poisson input and general service time distribution as:

$$L = \rho + \frac{\rho^2 + \lambda^2 \sigma_s^2}{2(1-\rho)}$$

where $\rho = \lambda / \ell R$

and σ_s is the variance of the service time distribution. [Ref. 8]
By applying Little's formula [Ref. 24]

$$L = \lambda W$$

it follows directly that the expected time to pass through the system, W, depends on the mean and variance of the service time distribution. In particular, if the variance of the actual service time distribution in a system is greater than that of the exponential distribution, then the M/M/1 queueing model will give an expected job time which is less

than the actual expected job time. There is reason to believe that the service time distributions of computers are sometimes high variance distributions. Figure 2.1 shows an example of such service time (CPU time) statistics. While the statistics shown have the general shape of an exponential distribution, they have a very long tail which gives then a high variance.

There are, however, also studies in which the exponential service time assumption gave results that corresponded closely to actual system statistics. An example of this is the study of the Michigan Terminal System by Moore. [Ref. 29]

The assumption of an infinite buffer makes the system a no loss system. This assumption is realistic if the system has a buffer of such size that overload occurs with extremely small probability.

2.2 Model of a Store and Forward Communication Channel

The model used for a store and forward communication channel is also an M/M/1 queue. As such, basically the same type of assumptions are made as for the model of a computer. The discussions about the Poisson input and infinite buffer assumptions carry over almost directly. The service time assumptions need to be examined separately however. For the communication channel it is assumed that

1. The length of messages in bits is distributed as a negative exponential with mean $1/\mu_p$ for programs (computer inputs) and mean $1/\mu_r$ for results (computer outputs).

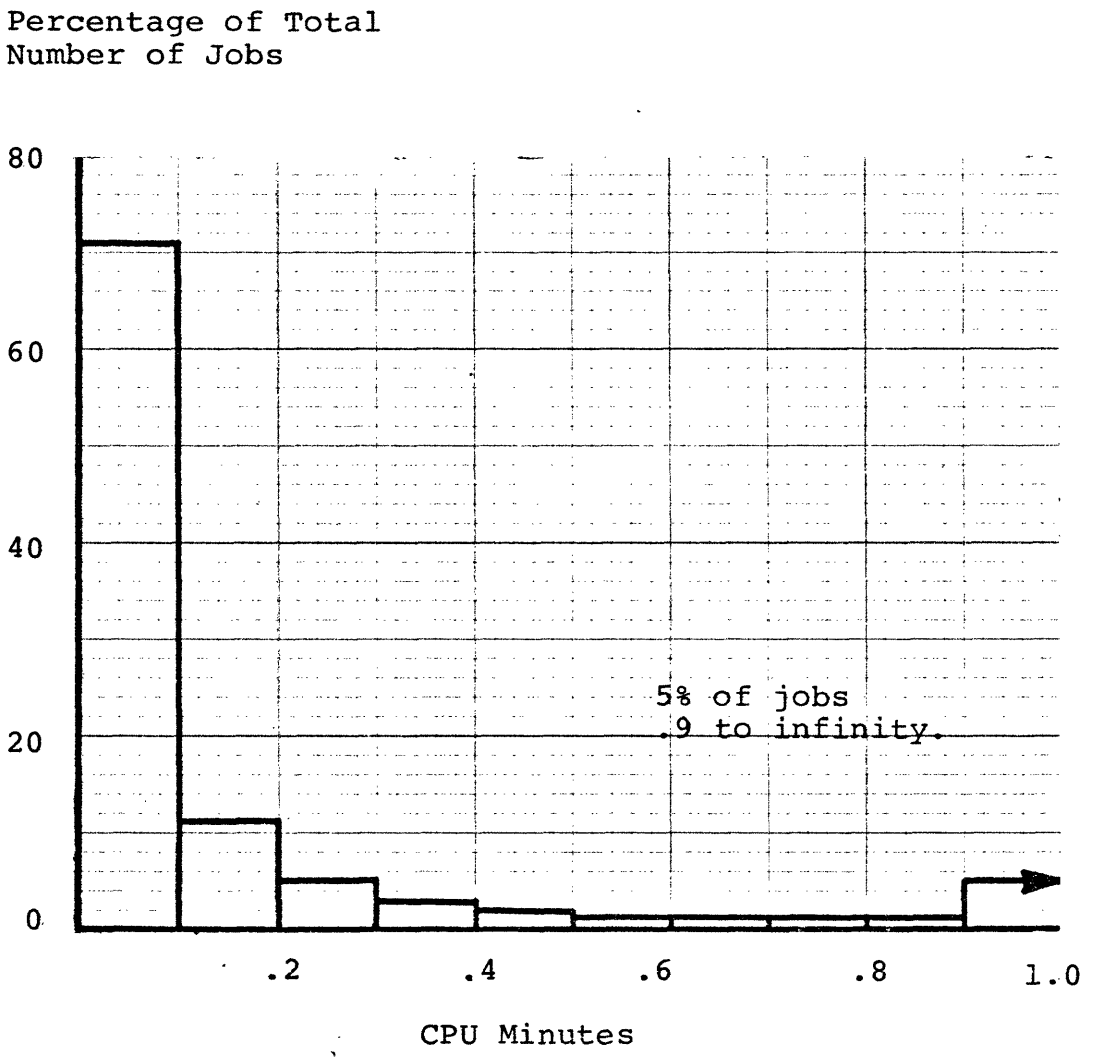


Figure 2.1 Sample Distribution of CPU Time per Computer Job. Source of data: M.I.T. Information Processing Center, Job Processing System Income Distribution Report, March 1975.

2. The communication channel has a capacity of C bits per unit time so that the time to transmit a message is also distributed as a negative exponential with mean $1/\mu_p C$ or $1/\mu_r C$.

3. If a message passes through more than one communication channel, its length is chosen independently at each channel through which it passes.

The queueing model of a store and forward communication channel has been extensively applied to the analysis of the ARPA Network by Kleinrock [Refs. 18 and 19]. In these studies the analytic queueing model with its exponential service time and independence assumptions gave an accurate representation of the basic performance characteristics of the network. However, in order to match the analytic results more closely to simulation results of network operation, it was found that the expression for average delay through the network needed to be modified to include delays other than those due to the finite time required to transmit a message over a finite bandwidth channel and those due to the resulting queueing. Delay terms to account for acknowledgement traffic, propagation delays and message processing delays were added to the analytic model. In this study those delay terms will be assumed to be zero. The result is that the system being analyzed is an idealized system in which communication delays result only from the limited capacities of communication channels and the associated queueing delays.

The independence assumption for messages that pass through more

than one communication channel is necessary to remove the statistical dependence between the interarrival times and message lengths of adjacent messages in the network. With this dependence, an analytic solution to the queueing network problem does not exist. This independence assumption has been studied in detail by Kleinrock [Ref. 17].

Another way in which the M/M/1 queue model is an idealization of actual implementations of message switched networks is that the model assumes that each message is transmitted as one block of data. In actual systems, long messages are often divided into packets, each of which can have its own routing through the network. The queueing theory for message delay when messages are divided into packets has been studied by Rubin [Ref. 32]. This study of load sharing assumes that messages are transmitted as one unit.

In modeling load sharing operation, it is important to examine the relationship between $1/\mu_p$ and $1/\mu_r$, the mean lengths of computer programs and computer results, respectively. There is evidence that the mean length of computer results is often an order of magnitude greater than the mean length of input programs. [Ref. 13] It is, however, also quite possible to think of systems where the input data and the output data are more nearly equal. An example is a control computer used to process many inputs to produce a single decision. In light of this, both the case of a ten to one and the case of a one to one ration of output to input will be investigated in the analysis that follows.

One final point that needs to be made is that the distribution of the message lengths of programs, the number of operations they require and the message length of results are assumed to be independent. While there is no physical basis for this assumption, it is required to make the problem mathematically manageable.

2.3 System Performance Measure

The measure used to evaluate the performance of a computer system in this study is the steady state expected time to process a computer job submitted to the system. The expression for this system expected job time is

$$
\begin{aligned}
E[T] &= \int_{\tau=0}^{\infty} \sum_{i=1}^{N} \tau \, P_T(\tau \,|\, \text{enter at } i) \, P_r(\text{enter at } i) \, d\tau \\
&= \sum_{i=1}^{N} P_r(\text{enter at } i) \int_{\tau=0}^{\infty} \tau \, P_T(\tau \,|\, \text{enter at } i) \, d\tau \\
&= \sum_{i=1}^{N} \frac{\lambda_i}{\lambda_T} E[T_i] \qquad (2.1)
\end{aligned}
$$

The time $E[T_i]$ is the expected time to process a computer job which enters the system at the i th computer. This expected time includes the computation time required by the job and, if the job is processed by a computer other than the one to which it was submitted, it also includes the communication time to send the program to the processing computer and the time to return the results to the point of origin.

The times $E[T_i]$ are weighted by the probability that an incoming job is submitted to the i th computer, which is the mean rate of jobs submitted to the i th computer (λ_i) divided by the total input rate to the system (λ_T). These terms are summed over all N computers in the system to give a system expected job time.

The next two sections examine upper and lower bounds of system performance based on expected job time.

2.4 Lower Bounds on System Performance

A lower bound† on the performance of a system of computers using load sharing is their performance without any load sharing. With each computer modeled as an M/M/1 queue with mean service time $1/\ell R_i$ the expression for $E[T_i]$ is (c.f. Appendix A)

$$E[T_i] = \frac{1}{\ell R_i - \lambda_i} \qquad 0 \leq \lambda_i < \ell R_i$$

Substituting into Equation 2.1 gives

$$E[T] = \frac{1}{\lambda_T} \sum_{i=1}^{N} \frac{\lambda_i}{\ell R_i - \lambda_i} \tag{2.2}$$

Figure 2.2 shows several plots of Equation 2.2 for a system of three

---

†Note that a lower bound on system performance is an upper bound on expected job time and that an upper bound on system performance is a lower bound on expected job time. Whenever the terms upper and lower bound are used in this study, they refer to system performance.

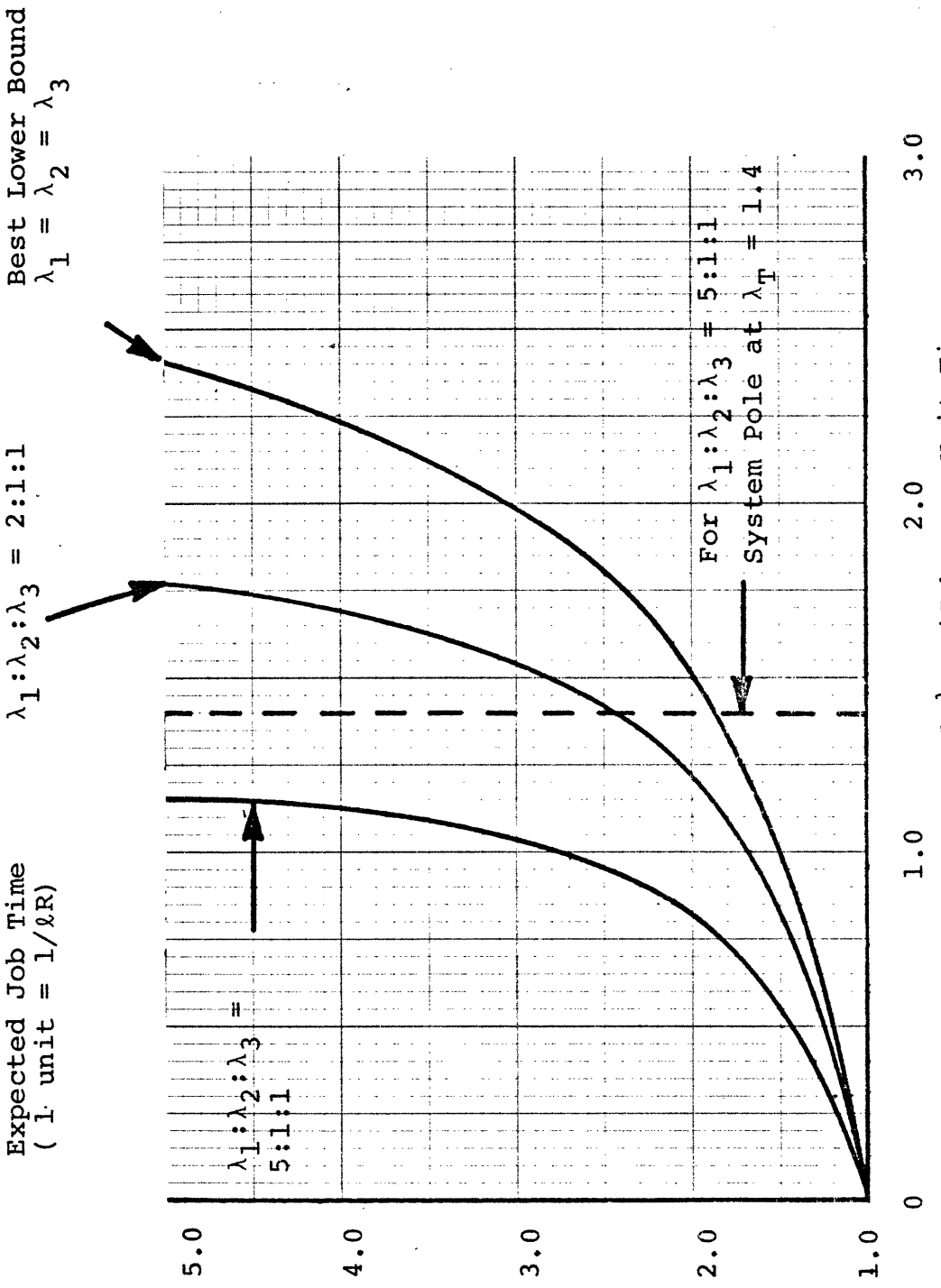


Figure 2.2 Lower Bounds on System Performance for a Three Computer System.

equal capacity computers.† The three curves shown in this graph are lower bounds on performance for three different load balances in the system. They illustrate the two basic characteristics of this lower bound function which are 1) that the function has a pole at λ_M; $0 \leq \lambda_M \leq \sum_{i=1}^{N} \ell R_i$ where N is the number of computers in the system and 2) that the more evenly the load is distributed in the system, the better the lower bound on performance.

The pole in the lower bound function occurs when one of the $\lambda_i = \ell R_i$. This is because for a steady state to exist at each of the computer queues, each λ_i must be less than ℓR_i. If $\lambda_i \geq \ell R_i$ then the queue at computer i becomes infinite and so does the waiting time, causing the system expected job time to be infinite as well. Figure 2.2 shows how a load imbalance can thus severely degrade system performance. In the case of a load imbalance where the ration $\lambda_1:\lambda_2:\lambda_3$ is 5:1:1, the system pole occurs when $\lambda_1 = 1$. The total system load at this point is only $\lambda_T = 1.4$. As will be analyzed in the next chapter, the correction of such degradation of system performance due to imblanaced loads is one of the main benefits of using load sharing in a computer-communication network.

For the case of equal capacity computers, the best lower bound on performance is achieved when the computers are equally loaded. In general, the best lower bound can be found by minimizing the expression

---

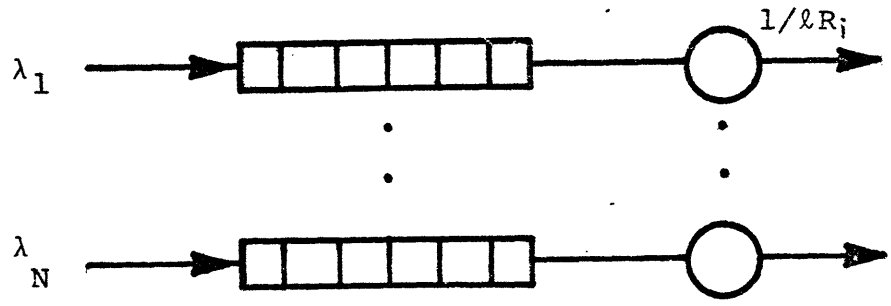
†In this figure, $1/\ell R$ is taken to be one unit of time. This convention will be followed throughout this study whenever all computers in the system have the same processing rate.

for expected job time (Equation 2.2) with respect to λ_i subject to the constraint $\sum_{i=1}^{N} \lambda_i = \lambda_T$ by using Lagrange multipliers.

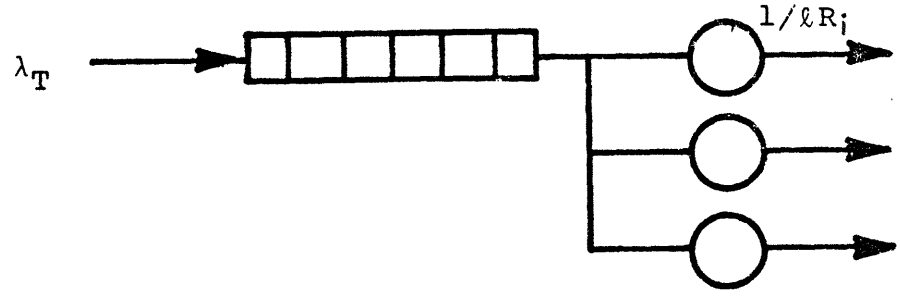
2.5 Upper Bounds on System Performance

Upper bounds on system performance for a system of computers can be obtained by using bounding models which represent the best possible use of the computing resources available. With each computer resource modeled as a single server with an exponential service time distribution of mean $1/\ell R_i$, there are two possible upper bound models, depending on the type of system operation one allows. The first bounding model is a multiserver queue with N servers, each with an exponential service time distribution of mean $1/\ell R_i$. The second bounding model is a single server queue with an exponential service time distribution of mean $1/\sum_{i=1}^{N} \ell R_i$. These two upper bound models, along with the lower bound model, are shown schematically in Figure 2.3.

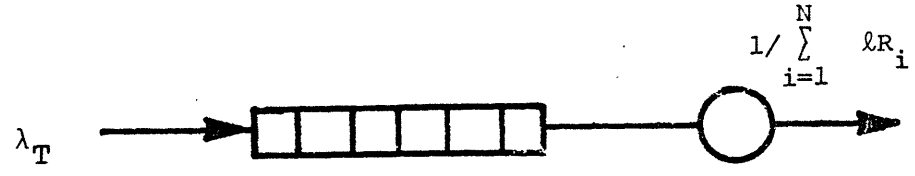
The multiserver queue bounding model assumes that all jobs arriving in the system are served in a first-come-first-served manner. Their service starts instantaneously using the largest capacity computer available in the system. Each job is processes by only one computer at a time, but whenever a job leaves the system, the remaining jobs are reassigned so that the largest capacity computers are always the ones being used. If all computers are busy, jobs are queued up in order and the service of each job in turn starts as soon as a computer becomes



Lower Bound Model: N Independent M/M/1 Queues.



Upper Bound Model Assuming No Parallel Processing: M/M/N Queue.



Upper Bound Model Assuming Parallel Processing:
M/M/1 Queue With Mean Service Time $1/N\ell R$

Figure 2.3 Bounding Models

available for it. This model is an upper bound model in that it assumes there exists an infinite capacity (zero delay) communication network for sending programs and results from one computer to another if a job is processed by a computer other than the one to which it was submitted. It also assumes that there is a global controller in the system which instantaneously makes job assignments.

The single server upper bound model also assumes an infinite capacity communication network and an instantaneous global controller. The difference in operation between it and the multiserver bounding model is that the single server runs only one job at a time. In order for a distributed computer system to operate like this single server, each computer job that enters the system must be divided into N parts which are processes in parallel using all N computers in the system at the same time. In this way each job would be run at a rate defined by the combined capacity of the computers in the system.

The expected time to pass through either of the upper bound models is the same performance measure as system expected job time. If all computers have the same capacity, the expected time to pass through the multiserver queue is given by

$$E[T] = \frac{P_o(\lambda_T/\ell R)^N (\lambda_T/N\ell R)}{N!(1 - \lambda_T/N\ell R)^2 \lambda_T} + 1/\ell R$$

$$\text{where } P_o = \left[\sum_{n=o}^{N-1} \frac{(\lambda_T/\ell R)^n}{n!} + \frac{(\lambda_T/\ell R)^N}{N!} \quad \frac{1}{1 - \lambda_T/N\ell R} \right]^{-1}$$

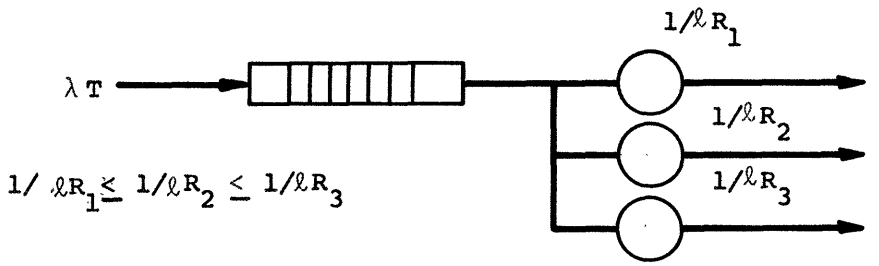
$$o \leq \lambda_T < N\ell R$$

as given in Appendix A. The expected time to pass through the single server queue is

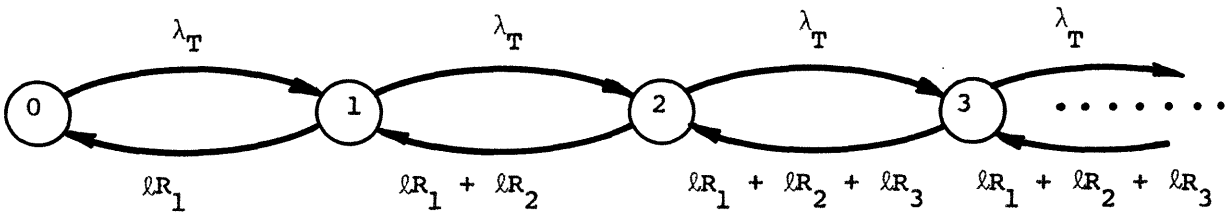
$$E[T] = \frac{1}{\sum_{i=1}^{N} \ell R_i - \lambda_T} \qquad o \leq \lambda_T < \sum_{i=1}^{N} \ell R_i$$

If one is interested in a network of computers of unequal capacity, the expected job time for the multiserver queue bounding model can be derived as follows. Since the arrival stream of jobs is Poisson, all job lengths are exponentially distributed and jobs are always processed on the largest capacity computer available, the system can be represented by a Markovian State diagram [Ref. 10] as shown in Figure 2.4. The states for this system are the number of customers in the system. The stochastic differential equations which describe the dynamics of these states are

$$\frac{\partial P_o(t)}{\partial t} = - \lambda_T P_o(t) + \mu_1 P_1(t)$$



a.) A 3 Computer Example



b.) The Corresponding State Transition Rate Diagram

Figure 2.4 The Multiserver Upper Performance Bound for a System of Computers With Unequal Capacities

$$\frac{\partial P_1(t)}{\partial t} = \lambda_T P_o(t) - \mu_1 P_1(t) + \mu_2 P_2(t)$$

$$\frac{\partial P_2(t)}{\partial t} = \lambda_T P_1(t) - (\lambda+\mu_2) P_2(t) + \mu_3 P_3(t)$$

$$\frac{\partial P_n(t)}{\partial t} = \lambda_T P_{n-1}(t) - (\lambda+\mu_3) P_n(t) + \mu_3 P_{n+1}(t)$$

$$n = 3, 4, 5 \ldots$$

where $P_n(t) = P_r$[system in state n at time t] and $\mu_1 = \ell R_1$; $\mu_2 = \ell R + \ell R_2$; $\mu_3 = \ell R_1 + \ell R_2 + \ell R_3$.

Since a steady state result is desired, the above equations are solved with $\frac{\partial P_n(t)}{\partial t} = 0$ for all n, in order to obtain recursive relationships between the steady state occupancy probabilities P_n. Using these recursive relationships it is possible to write all P_n as a function of P_o and one can then apply the requirement that

$$\sum_{n=o}^{\infty} P_n = 1$$

in order to solve for P_o. Once one has solved for all P_n in this manner, the expected time to pass through the queue (W) can be found by first calculating the expected number of jobs (L) in the system

$$L = \sum_{n=o}^{\infty} n P_n$$

and then applying Little's formula $L = \lambda_T W$. [Ref. 24)

Figure 2.5 shows a plot of the two upper bounds for a system of three equal capacity computers along with the best lower bound on system performance. Note that the expected job time for the multi-server queue at $\lambda_T = 0$ is $1/\ell R$, the same as for an independently operating system of computers. For the single server queue, however, the expected job time at $\lambda_T = 0$ is $1/N\ell R$. The single server model gives this better performance because whenever any job enters the system, all the computer resources are used to process it. In the multi-server model, if there are less than N jobs in the system, part of the computer resources are not being used. In a study of resource sharing, Kleinrock [Ref. 21] has shown that if it is possible to combine all system resources into a single server, this gives the best performance achievable with those resources. An important question to consider is if the combining of the computer resources of separate computers, as envisioned by the single server queue bounding model, is feasible. As stated before, in order for a distributed computer system to operate as efficiently as a single server queue, programs must be divided into N parts which can be processed in parallel by separate computers. Because of the many difficulties in achieving this sort of system operation,

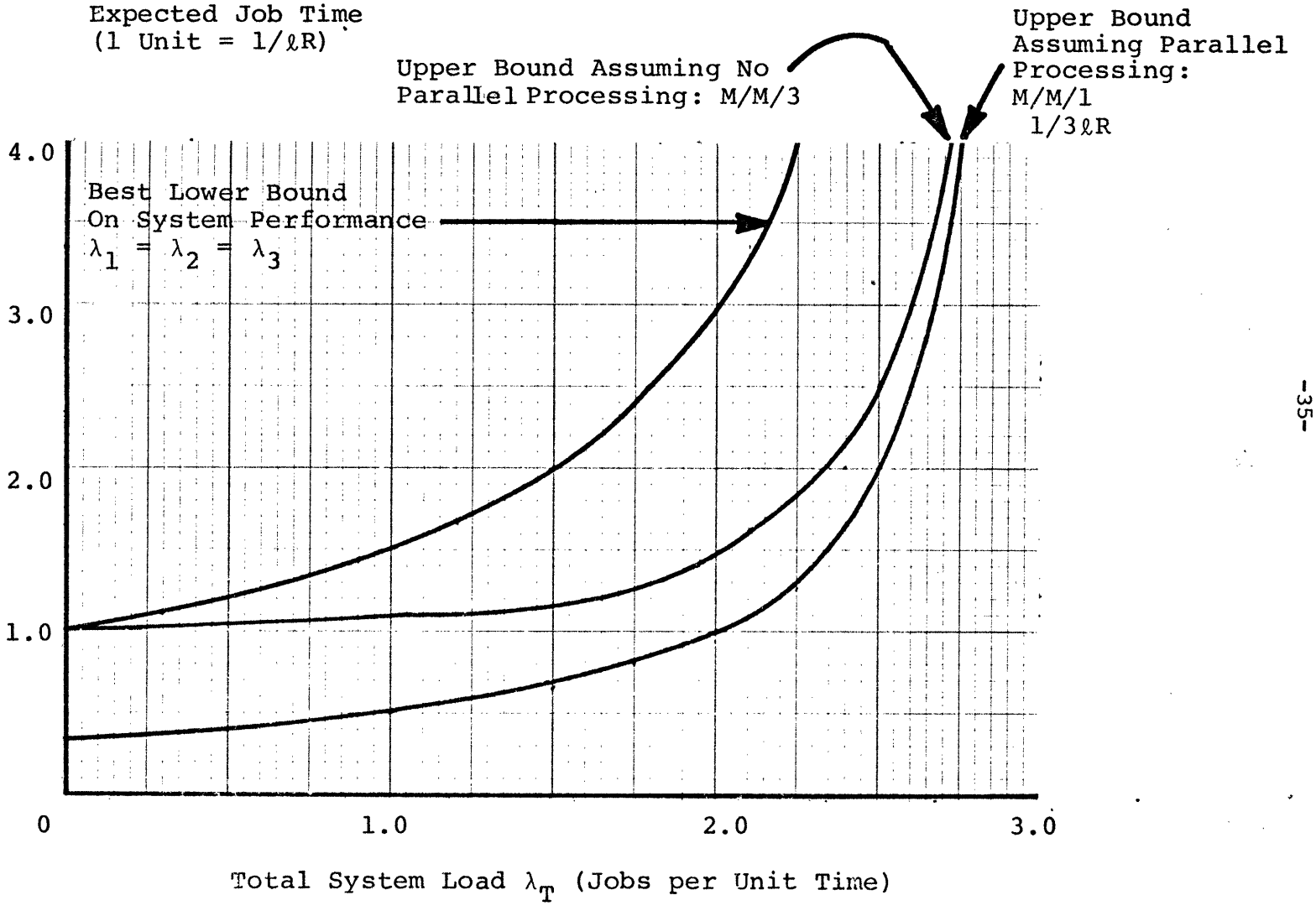


Figure 2.5 Upper Bounds on System Performance for a Three Computer System

the upper bound on system performance will be taken to be the performance of the multiserver queue model.

It is of interest to examine this upper bound on performance as a function of system size. Figure 2.6 illustrates the upper bound for various size computer systems consisting of equal capacity computers. The bound is plotted as a function of system utilization factor, $\rho = \lambda_T/N\ell R$. Also plotted is the best lower bound for all of the systems. The best lower bound, plotted as a function of system utilization factor, does not vary with system size.

In Figure 2.6 it can be seen that the upper bound improves with system size. The amount of improvement is greatest in going from small systems to medium size systems and decreases as a function of system size. As an example, if one considers going from a system of two computers operating at a utilization factor of 0.7 to a system of ten computers operating at a utilization factor of 0.7, the gain in the bound on expected job time is approximately $0.8/\ell R$. In going from a system of ten computers to an infinitely large system, also operating at a utilization factor of 0.7, the gain in the bound on expected job time is less than $0.1/\ell R$. This suggests that, unless the system is to be operated at an extremely high utilization factor, there may be little to be gained in terms of expected job time by increasing the size of the system beyond about ten to twenty computers. An important point to remember, however, is that the multiserver bounding model assumes a Poisson input stream of

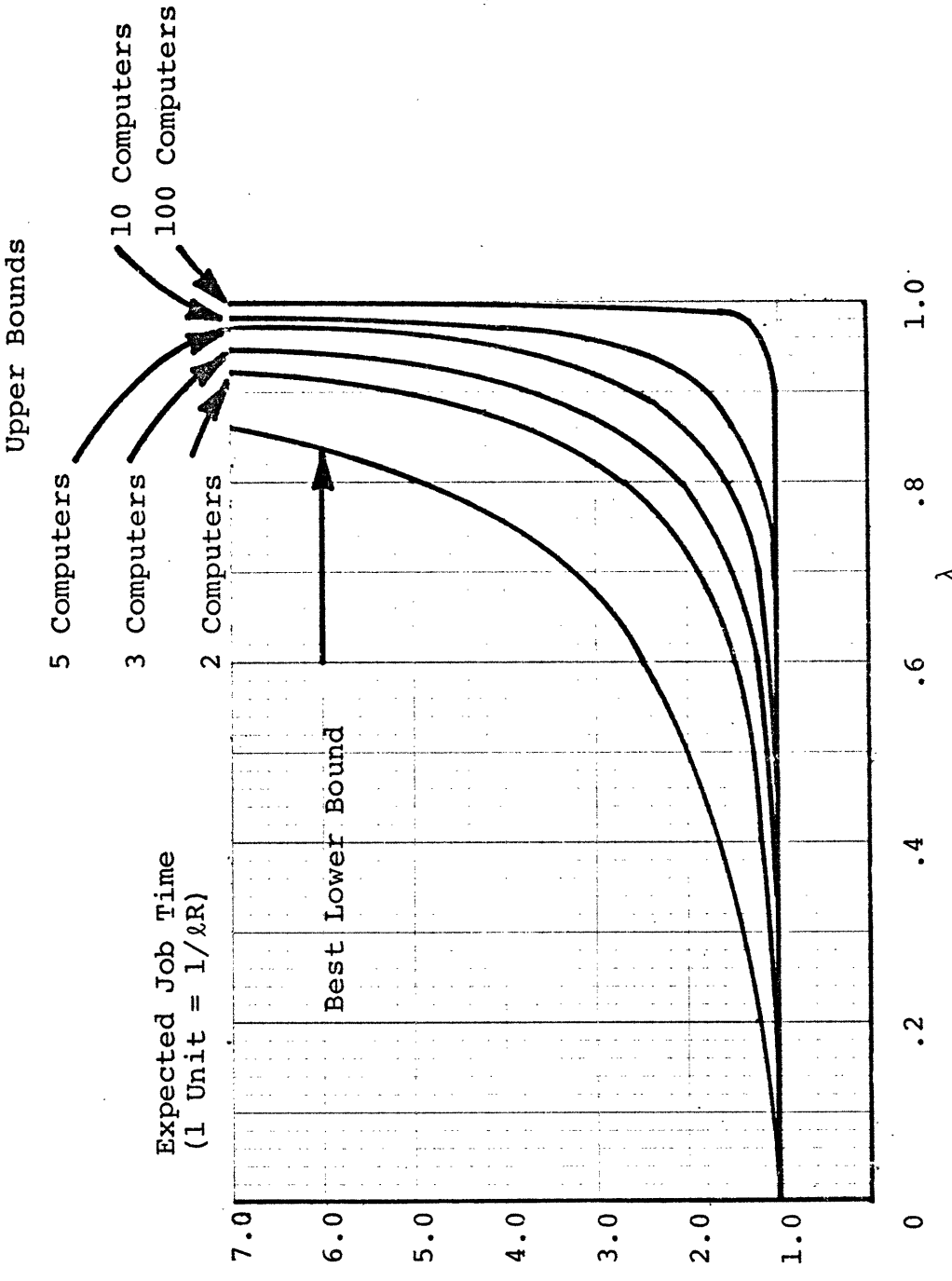


Figure 2.6 Upper Bounds on System Performance for Various Size Systems. Adapted from Ref. 25.

jobs. This means that the Poisson input streams of the individual computers which are combined into the input stream of the upper bound model must be independent of each other. If this does not hold in the system under consideration, the upper bound on system performance may not be as good as predicted by the multiserver queue model. In this study the required independence is assumed to exist.

With the upper and lower bounds on system performance that have been developed, one can identify the benefits in terms of expected job time that can possible be provided by load sharing in a computer-communication network. There are two general regions of improvement as depicted in Figure 2.7. The first region of improvement, region A, is the region between the lower bound on performance for an unbalanced load system and the lower bound for a balanced load system. Given an initially unbalanced load, operation in region A can be achieved by simply sending a fraction of the jobs arriving at the overloaded computers to the under loaded computers in the system. Which jobs to send can be determined by random sampling. This technique for load sharing will be called statistical load sharing. It is analyzed in detail in Chapter 3.

The second region of load sharing operation, region B, is the region between the best lower bound and the upper bound. Starting with a load balanced system of computers, it is necessary in order to achieve operation in this region that job assignment to computers be on the basis of the system state, i.e. which computers are available at the time of

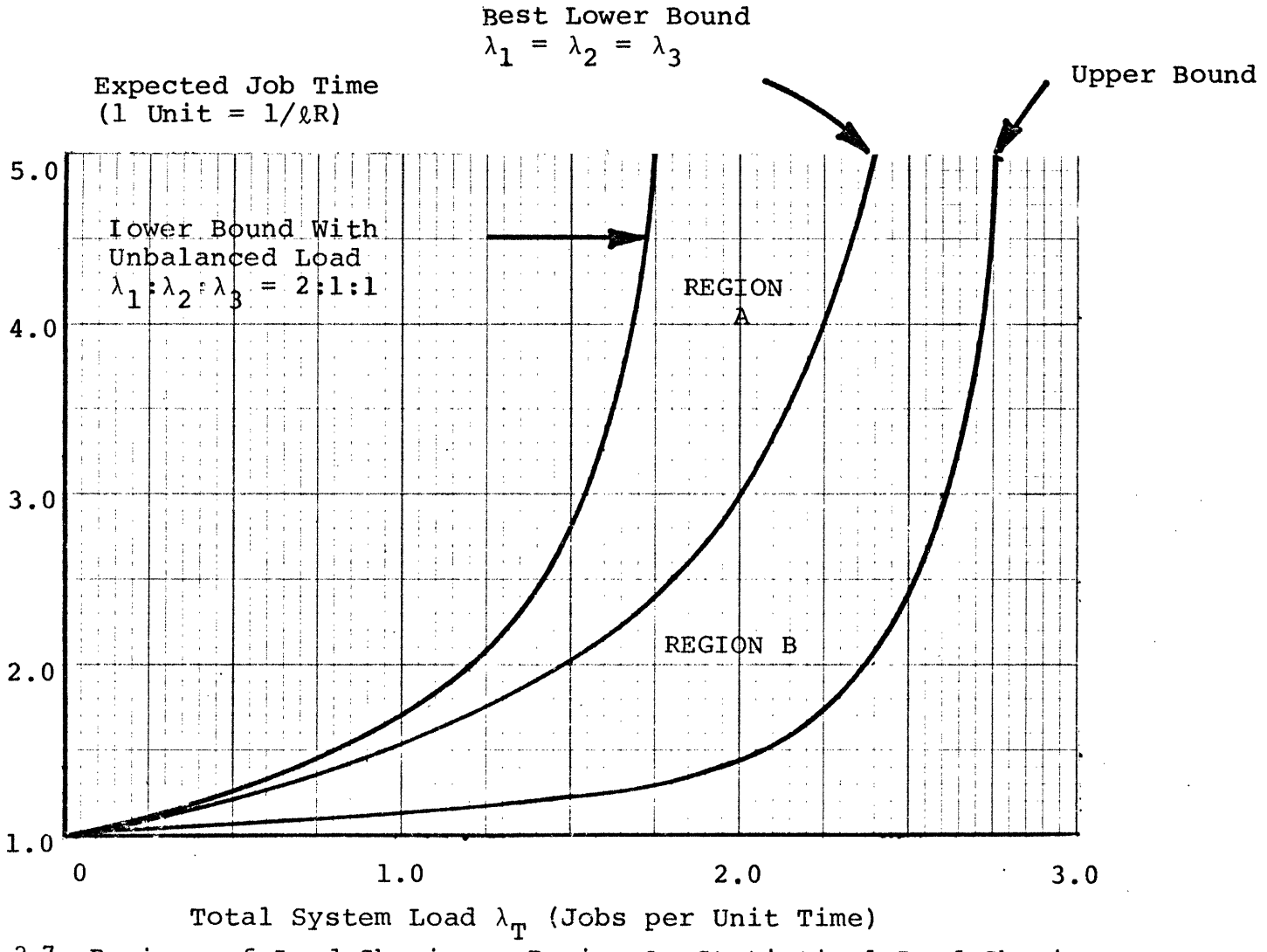


Figure 2.7 Regions of Load Sharing. Region A: Statistical Load Sharing. Region B: Dyanmic Load Sharing. Three computer example.

the assignment. A random decision rule such as statistical load sharing cannot improve system performance beyond the best lower bound. The achievement of system operation in region B will be called dynamic load sharing. A limited technique for dynamic load sharing will be analyzed in Chapter 4.

CHAPTER III STATISTICAL LOAD SHARING

3.1 Network of Queues Model

In Chapter 2 it was shown that for a system of independently operating computers, system performance in terms of expected job time improves as the total system load is more evenly distributed among the computers. Therefore, if a system of computers is operating in an unbalanced load situation, there is the possibility of improving system performance by simply sending a fraction of the jobs that arrive at overloaded computers to underloaded computers, by random sampling, in order to balance the load. This technique of load sharing in a computer-communication network, called statistical load sharing, is examined in this chapter.

A typical example of statistical load sharing operation is shown in Figure 3.1. In this case, Computer 1 is loaded more heavily than Computers 2 and 3 and therefore a fraction of the jobs which arrive at Computer 1, 2β, are sent to be processed at the underloaded computers. In order to evaluate the expected job time in this example, one must be able to determine the steady state expected time to pass through each of the computer and communication queues. This can be done by applying the following result for a network of queues due to Jackson [Ref. 12].

The result derived by Jackson applies to a network of M queues in which

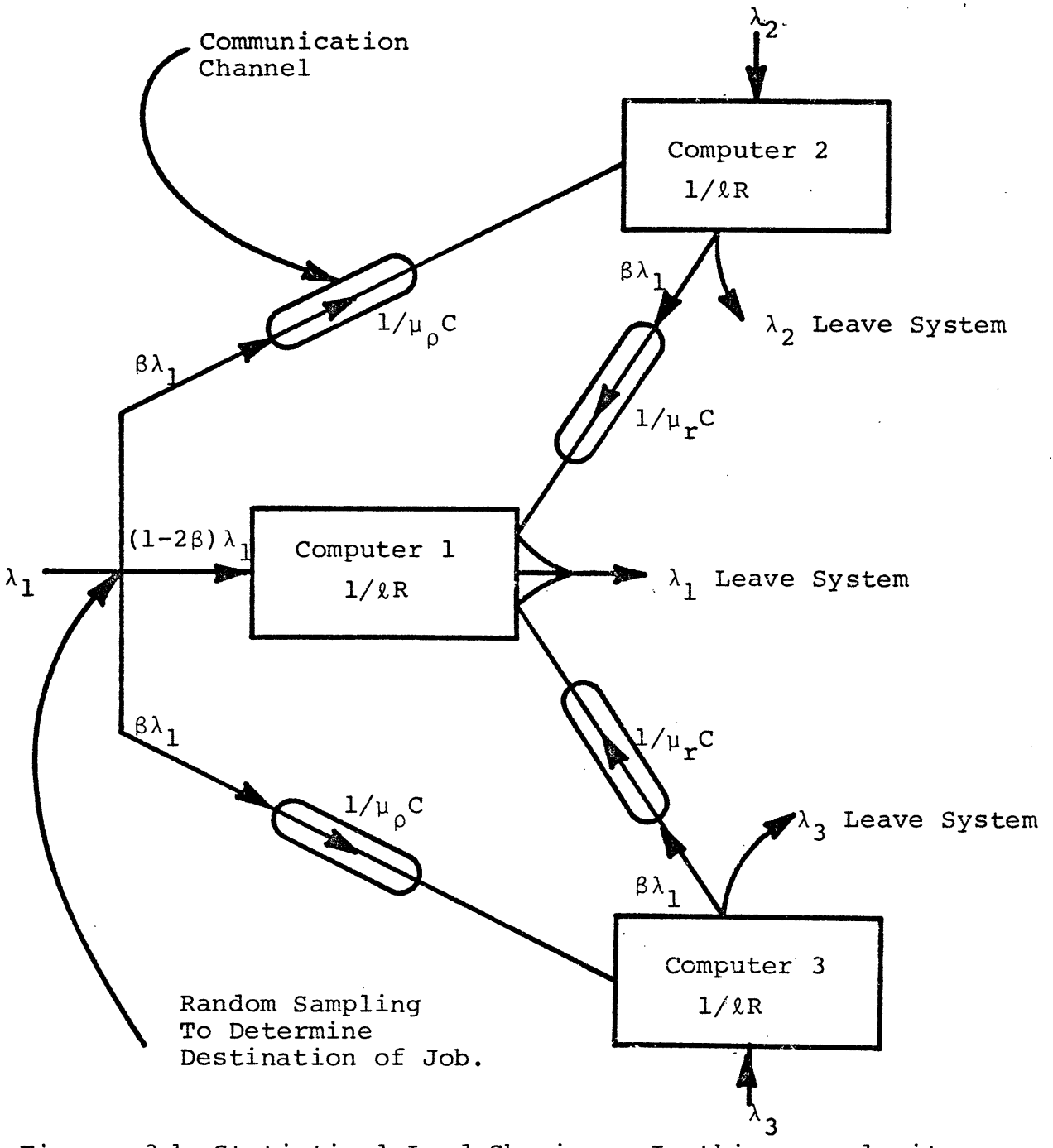


Figure 3.1 Statistical Load Sharing. In this example it is assumed $\lambda_1 > \lambda_2 = \lambda_3$.

1. Customers from outside the system arrive at each queue as a Poisson stream with mean rate λ_m.

2. Once served at queue m, a customer goes (instantaneously) to queue k (k=1,2,3. . . M) with probability θ_{km}. With probability

   $1 - \sum_{k=1}^{M} \theta_{km}$ the customer leaves the system.

3. Customers arriving at queue m (from inside or outside the system) are served in a first-come-first-served manner. The service time at each queue is distributed as a negative exponential with mean $1/\mu_m$.†

The network of computer and communication queues meets all of these requirements as discussed in Appendix B.††

For a network of queues as described above, let Γ_m (m=1,2,. . . M) be the average arrival rate of customers at stage m from inside and outside the system. Then in steady state, the following relations must hold

$$\Gamma_m = \lambda_m + \sum_{k=1}^{M} \theta_{mk} \quad K \qquad (m=1,2,3. . . M)$$

Now let K_m be the number of customers waiting and in service at queue m. The state of the system can then be defined as the vector

---

†Each queue can be a multiserver queue. In this analysis, however, it is assumed that each queue is a single server.

††If computer jobs must pass through more than one communication channel in succession, the independence assumption for communication service times discussed in Chapter 2 must be used.

$(K_1, K_2, \ldots K_M)$ and the following theorem due to Jackson holds.

> THEOREM. Define p_K^m $(m=1,2,\ldots M,\ K=0,1,2,\ldots)$ as the steady state probability of there being K customers in a M/M/1 queue with mean input rate Γ_m and mean service time $1/\mu_m$, i.e.
>
> $$P_K^m = (1 - \Gamma_m/\mu_m)(\Gamma_m/\mu_m)^K$$

Then the steady state distribution of the state of the above defined system is given by the products

$$P(K_1, K_2, \ldots K_M) = P_{K_1}^1 \; P_{K_2}^2 \ldots P_{K_M}^M$$

provided $\Gamma_m < \mu_m$ for $m = 1,2,\ldots M$.

This theorem states that in steady state, the system behaves as if the queues in the network were independent with inputs rates Γ_m. This result allows one to analyze the network of queues model for statistical load sharing by merely determining the mean input rates to each of the computer and communication queues.

In the next section, this approach is used to examine statistical load sharing in a fully connected symmetrical communication network with simple load imbalances. This is followed by a general formulation of the problem that applies to arbitrary communication topologies and load imbalances.

3.2 Analysis of Fully Connected Networks with Simple Load Imbalances

In this section, statistical load sharing in a fully connected symmetrical computer communication network with one computer overloaded and all others equally underloaded will be studied. Analysis of this special case allows one to gain insight into the basic operating characteristics of statistical load sharing. The approach used in this section is to start by considering a three computer system with a given load imbalance and analyzing it in detail. The operating characteristics observed will then be analyzed as a function of system load imbalance and as a function of system size.

A Three Computer Example

As a first example, consider a system of three computers in which Computer 1 is loaded twice as heavily as each of the other two computers ($\lambda_1:\lambda_2:\lambda_3 = 2:1:1$). The system is assumed to be symmetrical, i.e. all mean computer service times are equal as are all communication channel capacities. This results in basically the same situation as shown in Figure 3.1. In order to achieve statistical load sharing, some jobs arriving at Computer 1 are sent to Computers 2 and 3 for processing. Jobs are sent to Computer 2 with probability β and also to Computer 3 with probability β. With probability $1-2\beta$, jobs arriving at Computer 1 are processed there.

Using this load sharing strategy, the expression for system expected job time is

$$E[T] = \sum_{i=1}^{N} (\lambda_i/\lambda_T)\, E[T_i]$$

$$= \frac{\lambda_1}{\lambda_T} E[T_1] + \frac{\lambda_2}{\lambda_T} E[T_2] + \frac{\lambda_3}{\lambda_T} E[T_3]$$

$$= \frac{\lambda_1}{\lambda_T} \left[(1 - 2\beta)\left(\frac{1}{\ell R - (1- 2\beta)\, \lambda_1} \right) \right.$$

$$+ \beta\left(\frac{1}{\mu_p C - \lambda\beta_1} + \frac{1}{\ell R - (\lambda_2 + \beta\lambda_1)} + \frac{1}{\mu_r C - \beta\lambda_1}\right)$$

$$\left. + \beta\left(\frac{1}{\mu_p C - \beta\lambda_1} + \frac{1}{\ell R - (\lambda_3 + \beta\lambda_1)} + \frac{1}{\mu_r C - \beta\lambda_1}\right)\right]$$

$$+ \frac{\lambda_2}{\lambda_T}\left[\frac{1}{\ell R - (\lambda_2 + \beta\lambda_1)}\right] + \frac{\lambda_3}{\lambda_T}\left[\frac{1}{R - (\ _3 + \beta\lambda_1)}\right]$$

(3.1)

The expressions for the $E[T_i]$ are obtained by determining through which computer and communication queues a job must pass. Since in steady state each of these queues behave as if they were independent, the formula for the expected time to pass through an M/M/1 queue can be applied directly.

In this example, the mean average length of programs and results will be assumed to be equal $\frac{1}{\mu_r C} = \frac{1}{\mu_p C} = \frac{1}{\mu C}$. By also noting that λ_2 equals λ_3, Equation 3.1 can be simplified to

$$E[T] = \left[\frac{\lambda_1}{\lambda_T} \quad (1 - 2\beta) \left(\frac{1}{\ell R - (1 - 2\beta)\ \lambda_1} \right) \right.$$

$$\left. + \ 2\beta \left(\frac{2}{\mu C - \beta\lambda_1} + \frac{1}{\ell R - (\lambda_2 + \beta\lambda_1)} \right) \right]$$

$$+ \ \frac{2\lambda_2}{\lambda_T} \left[\frac{1}{\ell R - (\lambda_2 + \beta\lambda_1)} \right] \qquad (3.2)$$

For a given load level in the system, the β used for load sharing is the β which minimizes Equation 3.2. Figure 3.2 shows a graph of this value of β for several different mean communication channel service times. A graph of the associated values of system expected job time is shown in Figure 3.3. While the graph of expected job time shows the performance gains due to statistical load sharing, greater insight into the load sharing operation can be gained by examining the graph of β vs. system load first.

The graph of β vs. system load shows three basic characteristics of statistical load sharing operation. These are that 1) there is a threshold of load sharing operation, 2) using communication systems with sufficient capacity, there is an asymptote which the probability of sending a job approaches and 3) for inadequate communication systems, this asymptote is not reached. Each of these characterists will now be examined separately.

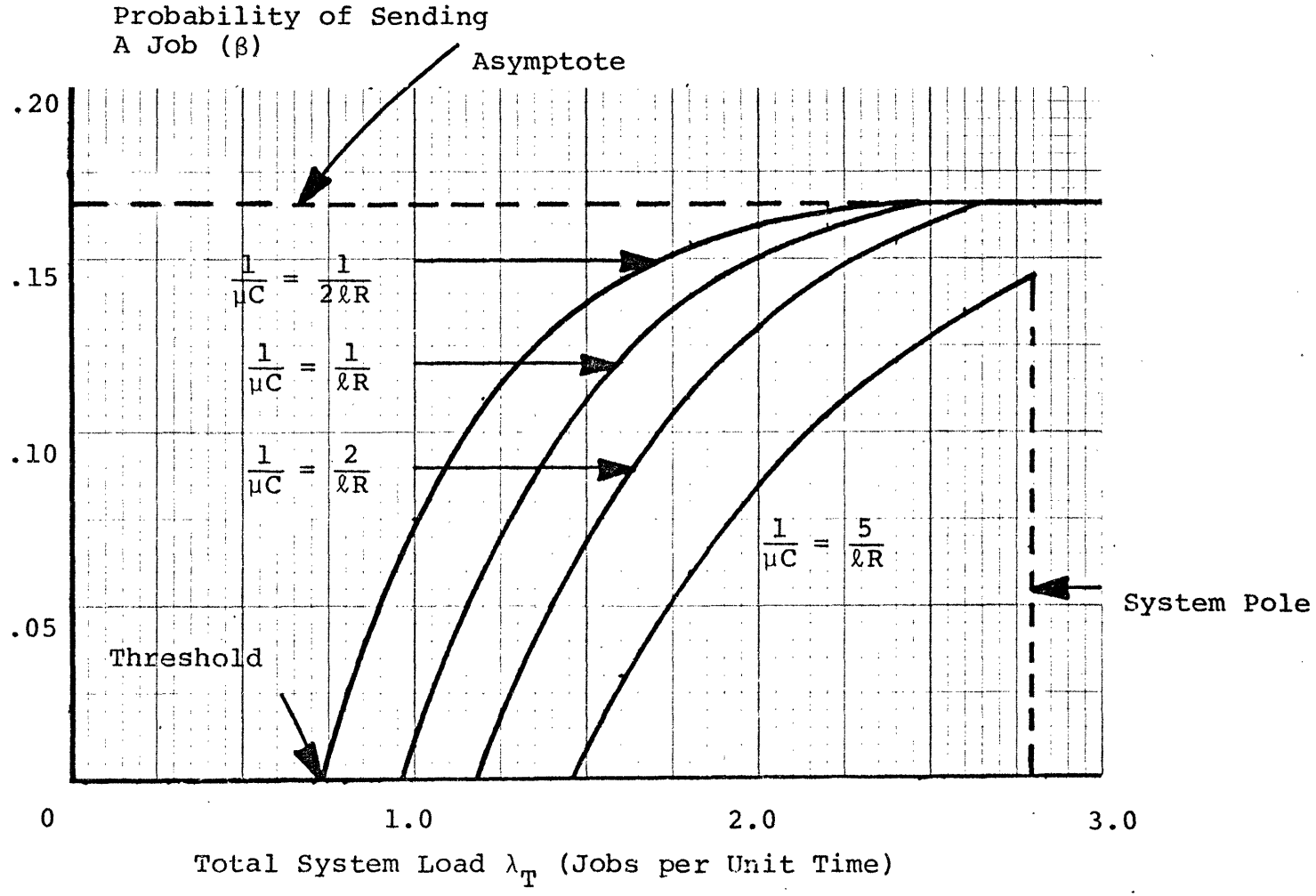


Figure 3.2 Probability of Sending a Job in a Three Computer System with $\lambda_1:\lambda_2:\lambda_3 = 2:1:1$.

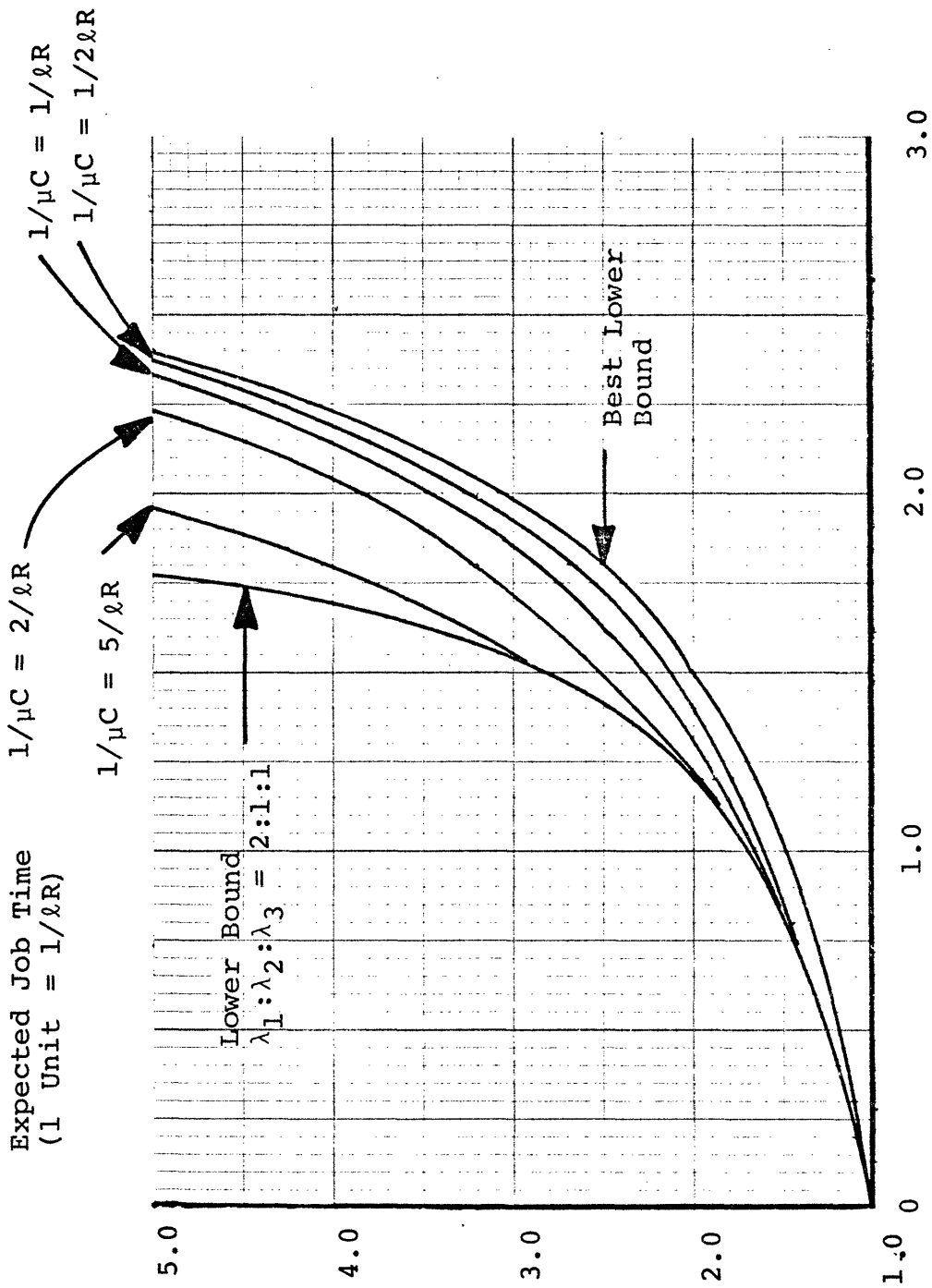


Figure 3.3 Expected Job Time in a Three Computer System with $\lambda_1:\lambda_2:\lambda_3 = 2:1:1$.

The Characteristic of Threshold

The threshold of load sharing operation occurs when the expected job time of a system of independently operating computers is equal to the expected job time of a system using load sharing in the limit as $\beta \to 0$. In this example, the expression for the expected job time of the system of independently operating computers is

$$E[T] = \sum_{i=1}^{N} \frac{\lambda_i}{\lambda_T} E[T_i]$$

$$= \frac{\lambda_1}{\lambda_T} \frac{1}{\ell R - \lambda_1} + \frac{\lambda_2}{\lambda_T} \frac{1}{\ell R - \lambda_2} + \frac{\lambda_3}{\lambda_T} \frac{1}{\ell R - \lambda_3}$$

(3.3)

The threshold condition for load sharing can be found by equating Equation 3.3 with the limit of Equation 3.2 as $\beta \to 0$. Rearranging terms and using the fact that $\lambda_2 = \lambda_3$ and that $\frac{1}{\mu_r C} = \frac{1}{\mu_p C} = \frac{1}{\mu C}$ gives

$$\lim_{\beta \to 0} \lambda_1 \left[\frac{1}{\ell R - \lambda_1} - \frac{1 - 2\beta}{1 - (1 - 2\beta)\lambda_1} \right] \Big/ 2\beta\lambda_1$$

$$+ 2\lambda_2 \left[\frac{1}{\ell R - \lambda_2} - \frac{1}{\ell R - (\lambda_2 + \beta_1)} \right] \Big/ 2\beta\lambda_1$$

$$= \lim_{\beta \to 0} \frac{2}{\mu C - \beta\lambda_1} + \frac{1}{\ell R - (\lambda_2 + \beta\lambda_1)}$$

(3.4)

Taking the limit of the right hand side of Equation 3.4 gives

$$\lim_{\beta \to 0} \lambda_1 \left[\frac{1}{\ell R - \lambda_1} - \frac{1 - 2\beta}{1 - (1 - 2\beta)\lambda_1} \right] \Big/ 2\beta\lambda_1$$

$$+ 2\lambda_2 \left[\frac{1}{\ell R - \lambda_2} - \frac{1}{\ell R - (\lambda_2 + \beta\lambda_1)} \right] \Big/ 2\beta\lambda_1$$

$$= \frac{2}{\mu C} + \frac{1}{\ell R + \lambda_2} \qquad (3.5)$$

The left hand side of Equation 3.5 is an approximate expression for the derivatives with respect to λ of the expected times to process jobs at each of the computers in the system. Therefore at threshold

$$\lambda_1 \frac{\partial \frac{1}{\ell R - \lambda_1}}{\partial \lambda_1} \approx \frac{2}{\mu C} + \frac{1}{\ell R - \lambda_2} + 2\lambda_2 \frac{\partial \frac{1}{\ell R - \lambda_2}}{\partial \lambda_2} \qquad (3.6)$$

Equation 3.6 states that the threshold of statistical load sharing operation occurs when the incremental decrease in expected job time at Computer 1 due to load sharing is equal to the incremental increase in expected job time at Computers 2 and 3, due to the jobs sent there from Computer 1, plus the expected time to process a job by sending it to another computer. The expected time to process at another computer is the expected communication time under no load conditions ($2/\mu C$) plus the expected job time at the other compuer $1/(\ell R - \lambda_2)$. The weighting

of the derivative terms is due to the form of the system expected value expression.

Note that the important characteristic of the communication system is not just the channel capacity, but the mean service time which is a function of both the channel capacity and the mean message length. For this reason, the cases studied are examples of various rations of mean communication channel service times ($1/\mu C$) to mean computer service time ($1/\ell R$).

The Characteristic of an Asymptote for the Probability (β) of Sending a Job

A second characteristic of statistical load sharing operation is that the optimum probability (β) of sending a job from the overloaded computer to each of the equally underloaded computers by random sampling sometimes approaches an asymptote. The asymptote is the value of which would distribute the load evenly in the system, since this is the condition that gives the best expected job time. This can be seen in the example under consideration by examining Equation 3.1. The asymptote is approached when the terms representing communication delay in Equation 3.1 are small with respect to the terms representing computation time. If this is the case

$$E[T] = \frac{\lambda_1}{\lambda_T}\left[\frac{1-2\beta}{\ell R-(1-2\beta)\lambda_1}+\frac{\beta}{\ell R-(\lambda_2+\beta\lambda_1)}+\frac{\beta}{\ell R-(\lambda_3+\beta\lambda_1)}\right]+\frac{\lambda_2}{\lambda_T}\left[\frac{1}{\ell R-(\lambda_2+\beta\lambda_1)}\right]+\frac{\lambda_3}{\lambda_T}\left[\frac{1}{\ell R-(\lambda_2+\beta\lambda_1)}\right]$$

$$= \frac{\lambda_1(1-2\beta)}{\lambda_T}\left[\frac{1}{\ell R-(1-2\beta)\lambda_1}\right]+\frac{\lambda_2+\beta\lambda_1}{\lambda_T}\left[\frac{1}{\ell R+(\lambda_2+\beta\lambda_1)}\right]+\frac{\lambda_3+\beta\lambda_1}{\lambda_T}\left[\frac{1}{\ell R+(\lambda_3+\beta\lambda_1)}\right] \qquad (3.7)$$

Equation 3.7 is exactly the expression for the expected job time of a system of independently operating computers with mean input rates $(1-2\beta)\lambda_1$, $\lambda_2+\beta\lambda_1$ and $\lambda_3+\beta\lambda_1$. Therefore, the best expected job time is obtained by choosing β so that the loads will be equal at all three computers. For this example this means

$$(1-2\beta)\lambda_1 = \lambda_2+\beta\lambda_1 \text{ and } \lambda_1 = 2\lambda_3$$

which gives $\beta = 1/6 = .167$ as shown in Figure 3.2

Figure 3.2 also shows that for some communication systems the asymptote is not approached. This occurs when the expression for system expected job time with the optimum β has a pole at $\lambda_T < N\ell R$. In physical terms this means that at the overloaded computer, the computer queue and all the associated communication queues used for load sharing saturate at a system load λ_T less than $N\ell R$. For the three computer system considered here, this occurs when a communication network with $1/\mu C = 5/\ell R$ is used. In this case the overloaded computer, Computer 1, can process $\ell R = 1$ job per unit time and there are communication facilities for sending another $2(.2\ell R) = 0.4$ jobs per unit time to be processed elsewhere. This means that all facilities at Computer 1 saturate at $\lambda_1 = 1.4$ or $\lambda_T = 2.8 < N\ell R = 3$.

Note that if the probability β of sending a job approaches the asymptote, the expression for expected job time using statistical load sharing does not have a pole at $\lambda_T < N\ell R$, whereas the expression for the expected job time of a system of independently operating computers with a load imbalance does have a pole at $\lambda_T < N\ell R$. This results in a significant gain in expected job time, when λ_T is large, by using statistical load sharing. More importantly, the statistical load sharing allows the system to operate at higher throughput rates with a load imbalance than is possible without load sharing. This can clearly be seen in Figure 3.3

Figure 3.3 shows that the expected job time using statistical load sharing improves with a decrease in mean communication service time, as one would expect. It also shows that for the system under consideration, a mean total communication time ($2/\mu C$) less than the mean computation time ($1/\ell R$) gives statistical load sharing performance that closely approaches the performance of a balanced load system.

The Case of Different Mean Lengths for Computer Input and Output

As discussed in Chapter 2, in some computer systems, the mean message length of results is an order of magnitude greater than the mean message length of programs (inputs). If this relationship is assumed to hold in the three computer system with $\lambda_1:\lambda_2:\lambda_3 = 2:1:1$, Equation 3.1 is still the expression for the expected job time, but now $1/\mu_r C = 10/\mu_p C$. Graphs of expected job time and the probability of sending a job for this case are shown in Figures 3.4 and 3.5 respectively. These graphs have the same general characteristics as those in the previous example. The main difference is that the communication delay incurred in the system is essentially all due to delay in returning the results to the computer of origin. Note particularly that when the system saturates at $\lambda_T < N\ell R$, it is the overloaded computer queue and the communication queues for returning the results that saturate.

It is of interest to examine statistical load sharing operation as a function of load imbalance and system size. Operation as a function of load imbalance will be examined first.

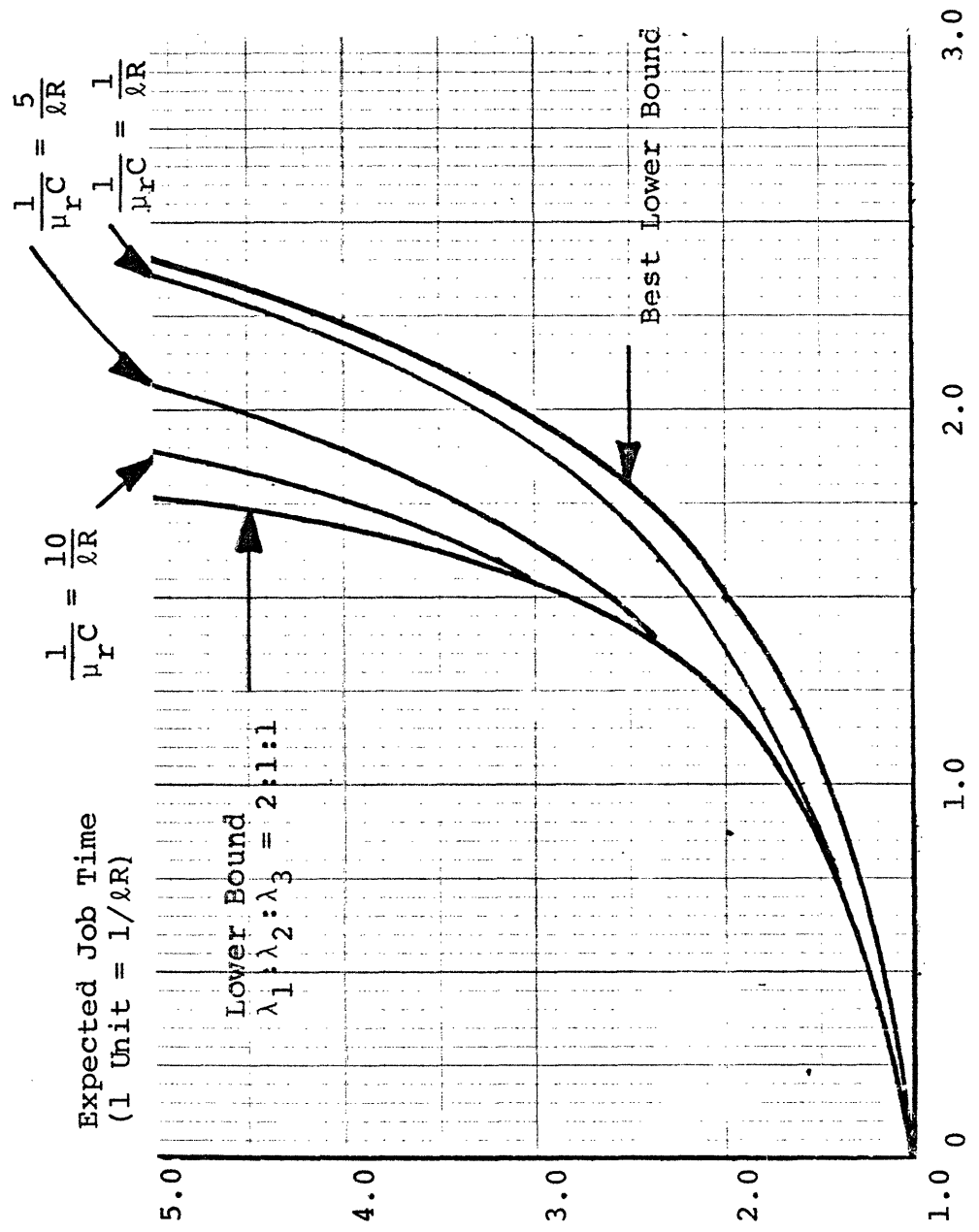


Figure 3.4 Expected Job Time in a Three Computer System with $\lambda_1:\lambda_2:\lambda_3 = 2:1:1$ and $\frac{1}{\mu_r C} = \frac{10}{\mu_p C}$.

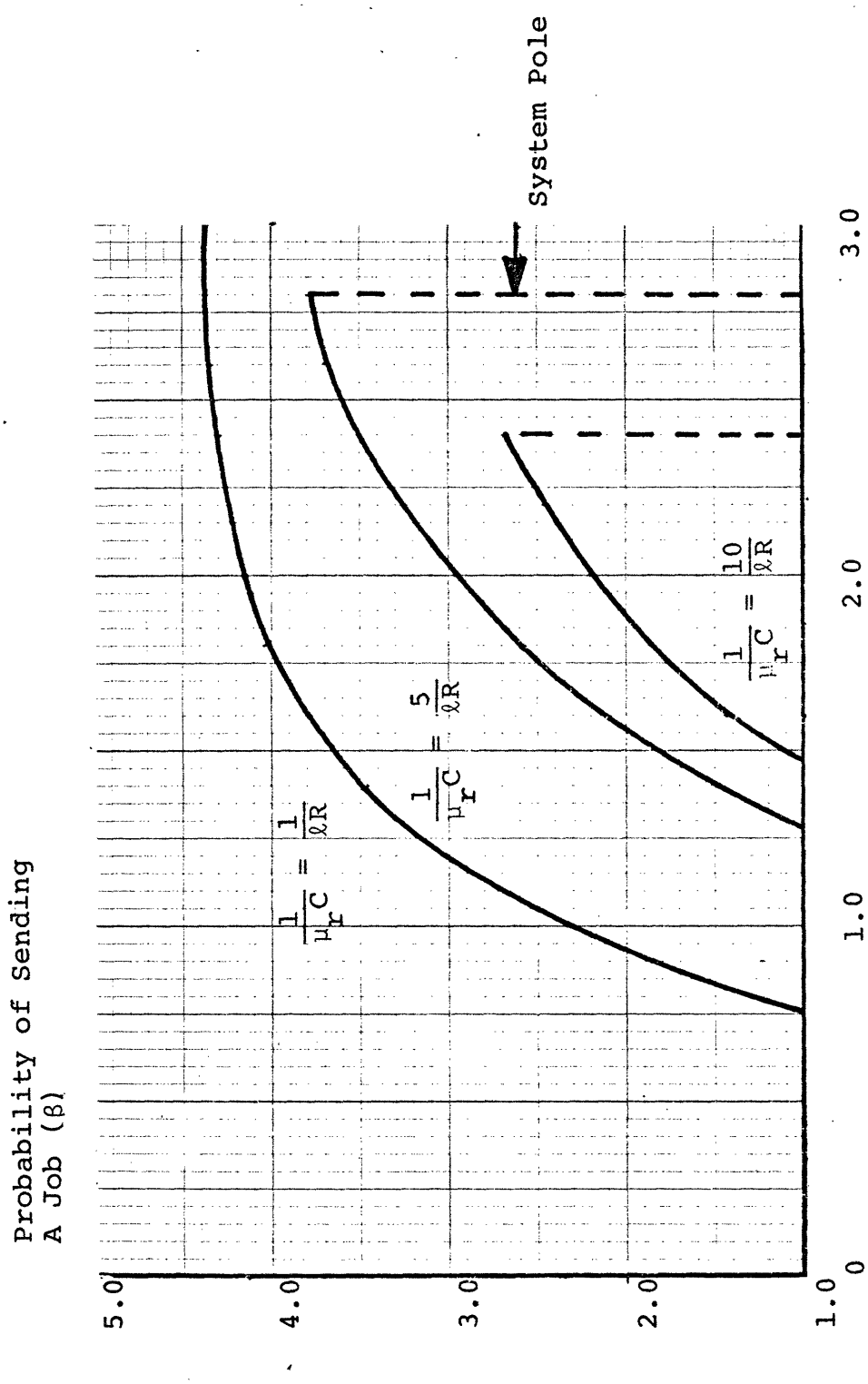


Figure 3.5 Probability of Sending a Job in a Three Computer System with $\lambda_1:\lambda_2:\lambda_3 = 2:1:1$ and $\frac{1}{\mu_r C} = \frac{10}{\mu_p C}$

Operating Characteristics as a Function of Load Imbalance

Consider a three computer system as before in which the load imbalance is $\lambda_1:\lambda_2:\lambda_3 = 4:1:1$. It is assumed that $1/\mu_r C = 1/\mu_p C = 1/\mu C$. Graphs of expected job time and the probability (β) of sending a job for this case are given in Figures 3.6 and 3.7 respectively. By comparing Figures 3.3 and 3.6, one can see that usually the greater the load imbalance in the system, the greater the range of system load λ_T over which statistical load sharing can provide improvement in expected job time. This can also be seen by comparing Figures 3.2 and 3.7 and noting that the thresholds of load sharing operation are lower in the more unbalanced system. One exception to the increase in the useful range of statistical load sharing with increased load imbalance is when the communication system is such that a pole occurs in the expression for expected job time at $\lambda_T < N\ell R$. If this is the case, the greater the load imbalance, the smaller the value of λ_T at which the system saturates. This can be seen by examining the case of $1/\mu C = 5/\ell R$ in Figures 3.2 and 3.7.

Operating Characteristics as a Function of System Size

Statistical load sharing operation in a fully connected symmetrical computer-communication network will now be investigated as a function of a system size. Examples of five and ten computer systems will be examined. Figures 3.8 and 3.9 show graphs of the expected job time and

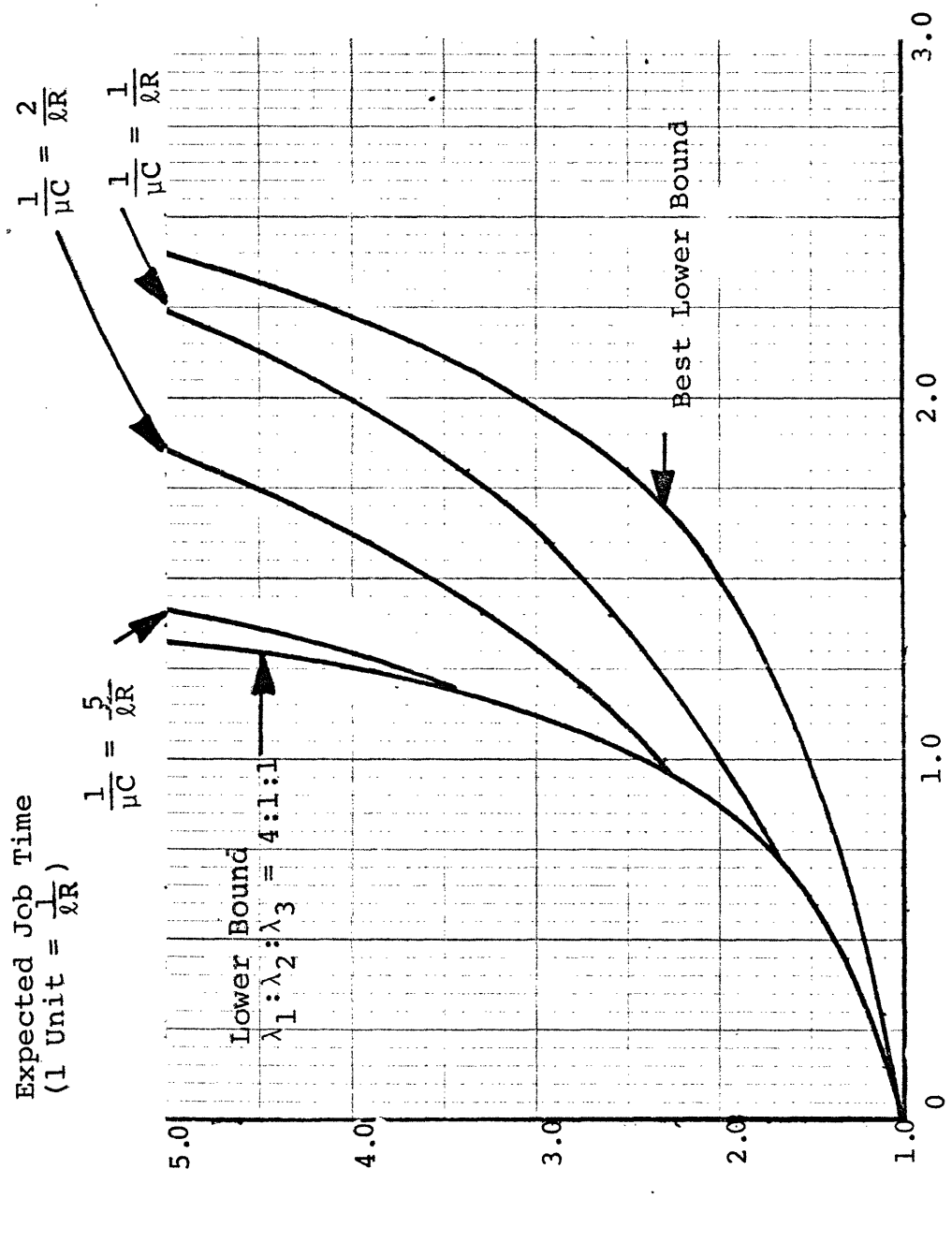


Figure 3.6 Expected Job Time in a Three Computer System with $\lambda_1:\lambda_2:\lambda_3 = 4:1:1$

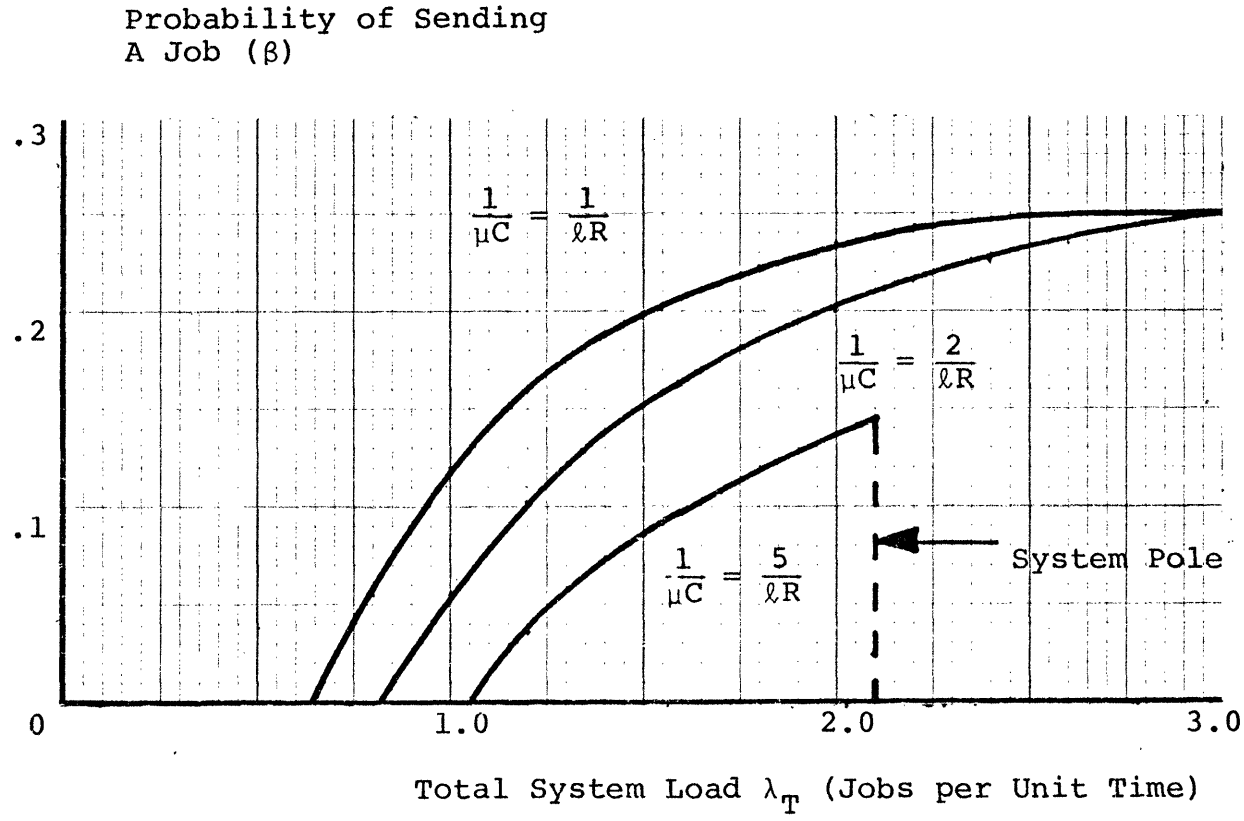


Figure 3.7 Probability of Sending a Job in a Three Computer System with $\lambda_1:\lambda_2:\lambda_3 = 4:1:1$

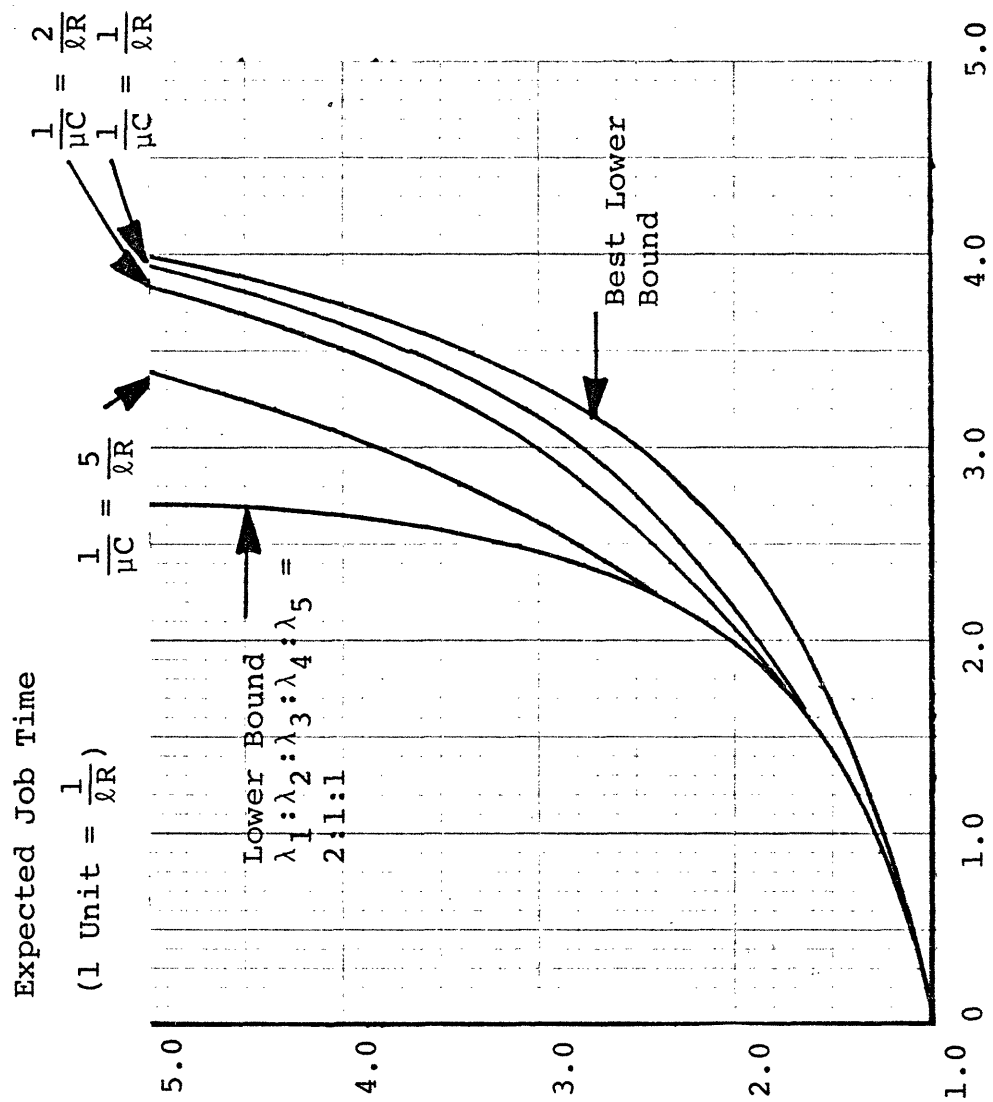


Figure 3.8 Expected Job Time in a Five Computer System with $\lambda_1:\lambda_2:\lambda_3:\lambda_4:\lambda_5$ = 2:1:1:1:1 .

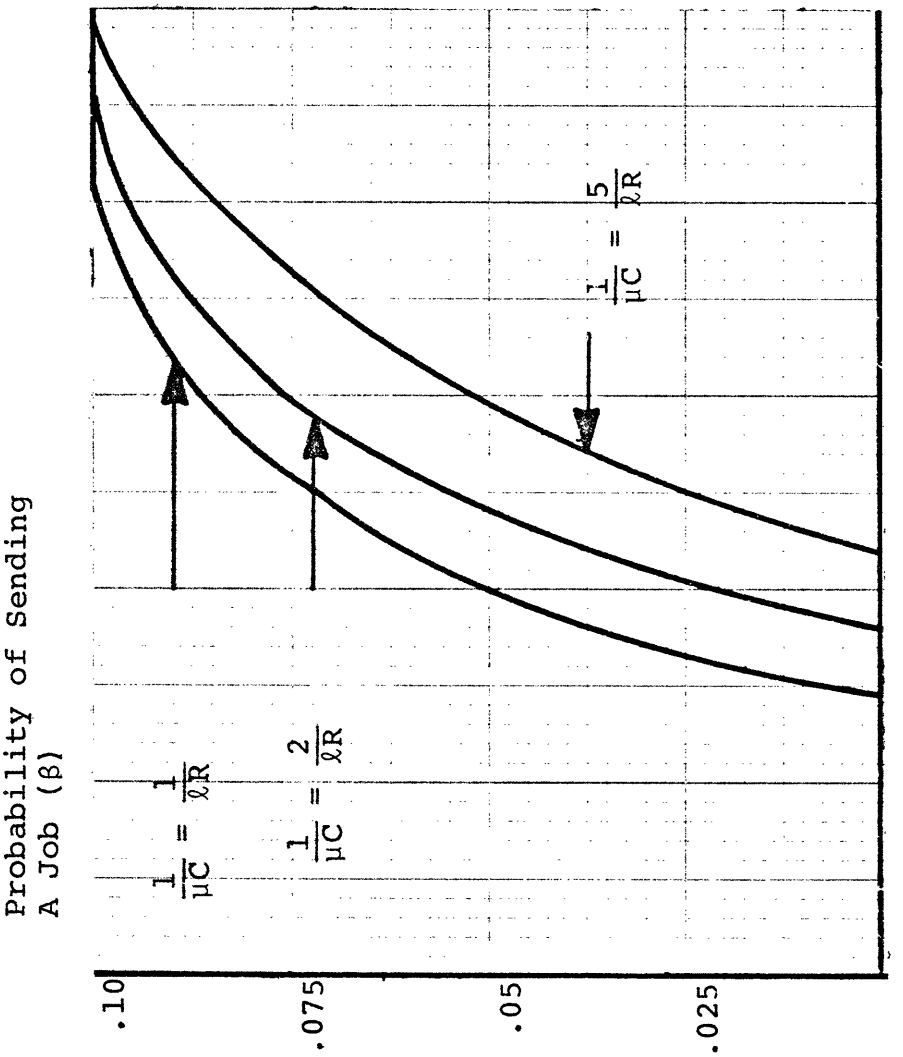


Figure 3.9 Probability of Sending a Job in a Five Computer System with $\lambda_1:\lambda_2:\lambda_3:\lambda_4:\lambda_5 = 2:1:1:1:1$.

the probability of sending a job in a five computer system with a load imbalance of $\lambda_1:\lambda_2:\lambda_3:\lambda_4:\lambda_5 = 2:1:1:1:1$. As in the three computer system, load sharing is accomplished by sending a fraction (β) of the jobs that arrive at Computer 1 to each of the other computers.

Figures 3.10 and 3.11 show graphs of the expected job time and the probability of sending a job in a ten computer system. In this system, Computer 1 is also loaded twice as heavily as each of the other computers in the system. Again load sharing is achieved by sending a fraction (β) of the jobs which arrive at Computer 1 to each of the other computers.

The main effect of system size is that as a system grows there are more computers to share the overload with and so a smaller fraction of jobs needs to be sent to each underloaded computer. As a result, the communication channels in a large system are less loaded, reducing the delay per job through them and improving system performance in terms of expected job time. In the examples considered here, the expected job time in the large system is further reduced by the fact that a smaller fraction of the jobs submitted to the system are submitted at the overloaded computer. As a result, even relatively slow communication networks, such as the case $1/\mu C = 5/\ell R$, provide performance that is quite good in a ten computer system.†

The increase in size of the system also improves the operation of statistical load sharing with slow communication networks by moving the

---

†It is important to note that the size of a fully connected communication network as considered here increases as N^2 where N is the number of computers (nodes) in the network.

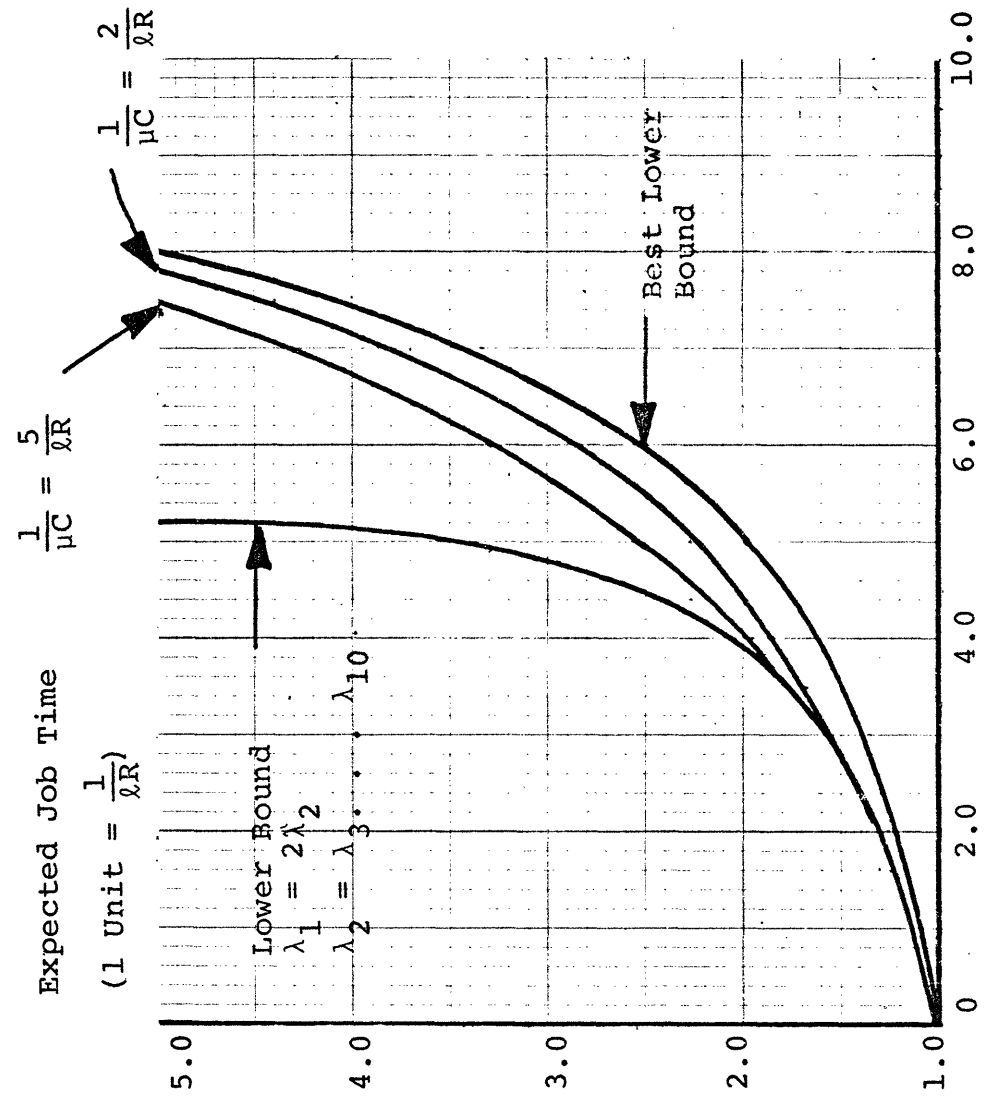


Figure 3.10 Expected Job Time in a Ten Computer System with $\lambda_1 : \lambda_i = 2:1$ (i=2,3, . . . 10).

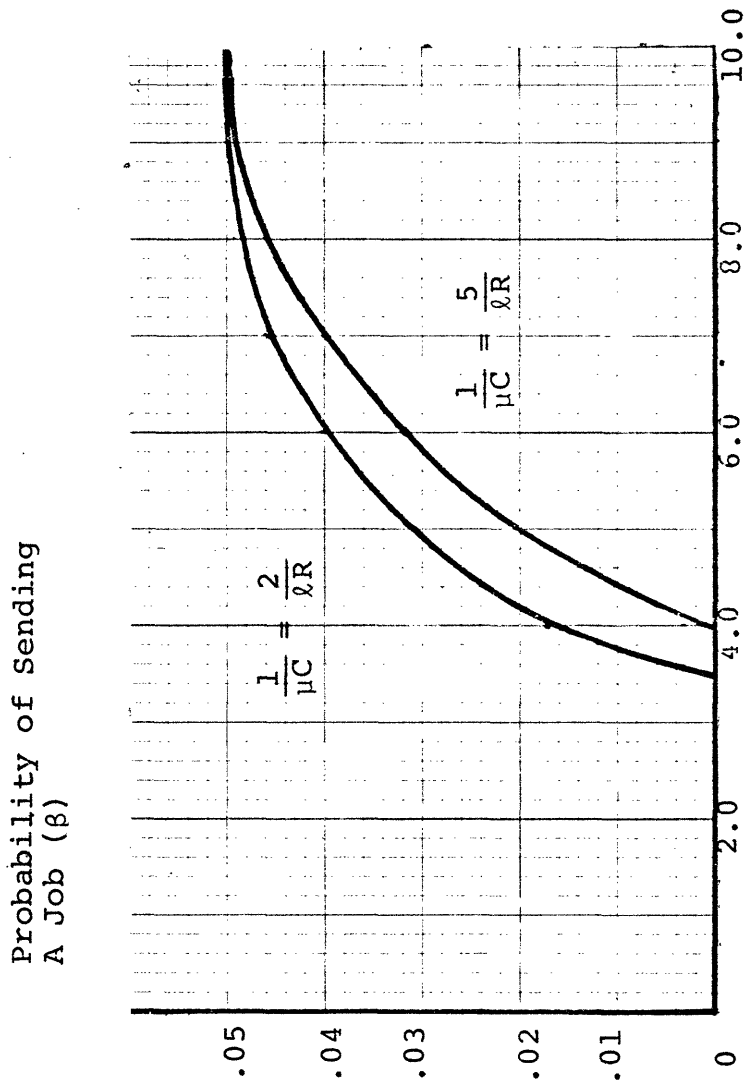


Figure 3.11 Probability of Sending a Job in a Ten Computer System with $\lambda_1 : \lambda_i = 2:1$ (i=2,3,. . . 10).

system pole to $\lambda_T = N\ell R$. This can be seen for the case $1/\mu C = 5/\ell R$. In the three computer case with a load imbalance of $\lambda_1:\lambda_2:\lambda_3 = 2:1:1$, such a communication network gave a system pole at $\lambda_T = 2.8 < N\ell R$. In the five computer case with the same load imbalance such a communication network ($1/\mu C = 5/\ell R$) does not give a system pole at $\lambda_T < N\ell R$. This is because as stated before, with more computers with which to load share, each communication channel in the fully connected network carries a smaller amount of traffic.

Summary of Operating Characteristics of Example Networks

Together, the previous examples have served to show some of the basic operating characteristics of statistical load sharing in a fully connected symmetrical computer-communication network with simple load imbalances. In summary these characteristics are

1. Statistical load sharing can provide significant improvement in expected job time performance for large system loads by correcting load imbalances. Most importantly, the system can operate at higher throughput levels than are possible without load sharing.

2. There exists a threshold of load sharing in the system which is a function of the mean communication time and the system load imblance.

3. There exists an asymptote that the optimum probability of sending a job approaches if the capacity of the communication network is such that the expression for system expected job time does not have a pole at $\lambda_T < N\ell R$. This asymptote is a function of system size and load imbalance.

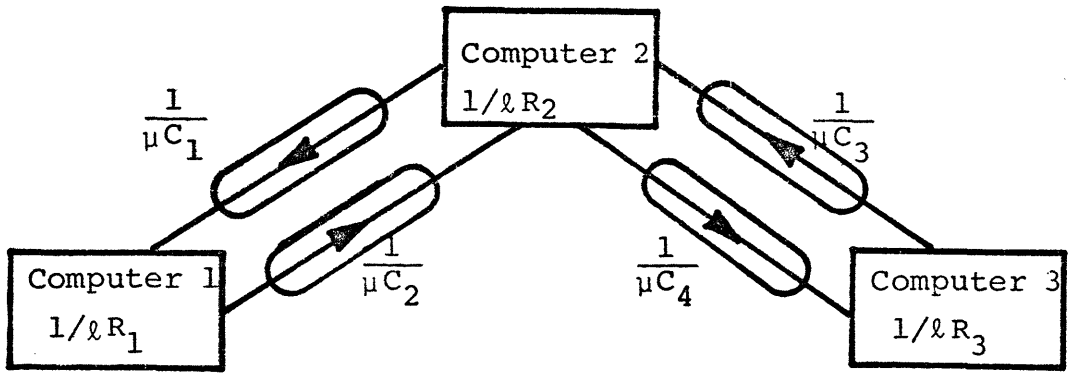
4. The asymptote will not be approached if the capacity of the communication network is such that the expression for system expected job time has a pole at $\lambda_T < N\ell R$.

While these summary points apply to the specific examples considered, similar characteristics will be found in the general case of statistical load sharing which will be formulated in the next section.

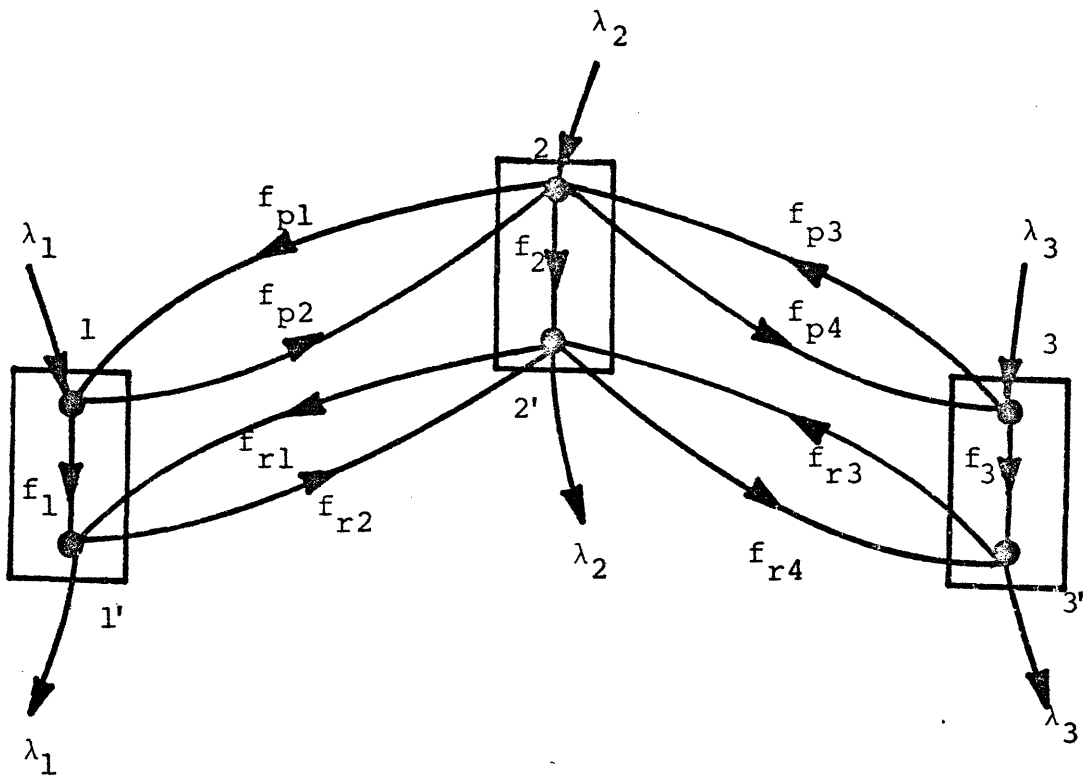
3.3 Formulation of the General Statistical Load Sharing Problem

In this section, the general statistical load sharing problem is shown to be a nonlinear multicommodity flow problem which can be solved by an efficient optimization technique. The examples of statistical load sharing given in the previous section were restricted to fully connected symmetrical computer-communication networks with simple load imbalances. The general formulation given here allows for arbitrary communication networks, arbitrary load imbalances and different rates of services (R_i) at each of the computers.

It is easiest to understand the equivalence between the statistical load sharing problem and a multicommodity flow problem by considering an example. Figure 3.12a shows a three computer system with a partially connected communication network. The logical flows of jobs that can occur in the system are shown in Figure 4.12b. Each computer is thought of as having two nodes, an input node and an output node, connected by a directed arc representing computer service. Jobs arrive at the input node of a computer at rate λ_i. There is a requirement that the jobs



a) Three Computer system with a partially connected communication network.



b) Logical flow of jobs in the network.

Figure 3.12 General Formulation of Statistical Load Sharing.

arriving at input node i receive computer service somewhere in the system and leave the system at output node i'. The jobs can be processed either at the computer to which they are submitted, or the programs can be sent to another computer input node. The possible flows of computer programs are represented by the flows f_{pj} in Figure 3.12b. The subscript j identifies the communication channel over which the actual flows would occur. If a job is processed at a computer other than the one at which it was submitted, the results must be sent back to the computer of origin. The possible flows of results are the flows f_{rj}, the subscript j again referring to the communication channel over which the actual flow occurs.

In order to assure that jobs submitted at input node i leave the system at output node i', all jobs are identified as to their origin. This means that the return route of a computer job is fixed once it has been processed. In the network of queues model, this fixed routing based on the job origin is equivalent to the random sampling used to determine the flows in the various routes through the network. As discussed in Appendix B, this is because the identification of jobs in a Poisson stream consisting of two Poisson substreams with different origins is equivalent to random sampling of the combined job stream.

In terms of the logical flows in the network, the statistical load sharing problem is now a multicommodity flow problem. The flows

required are for computer jobs to flow from node i to node i' at rate λ_i. Note that in the logical flow network, this requirement can be met only by having each computer job pass through a computer once and only once, as required by the load sharing problem.

The multicommodity flow problem is to determine the optimal flow through a network subject to a well defined convex objective function and a set of convex constraints. In the load sharing problem the objective function is expected job time. Assuming that the message lengths for programs and results are equal ($\mu_p = \mu_r = \mu$),† the expression for the expected time to pass through the logical flow network is

$$E[T] = \sum_{\text{all arcs}} E[T \text{ through arc } i] \Pr[\text{pass through arc } i]$$

$$= \frac{1}{\lambda_T} \sum_{i=1}^{N} \frac{f_i}{\ell R_i - f_i} + \sum_{j=1}^{NC} \frac{f_{pj} + f_{rj}}{\mu C_j - (f_{pj} + f_{rj})} \qquad (3.8)$$

where N = number of computers

NC = number of communication channels

$\lambda_T = \sum_{i=1}^{N} \lambda_i$ total input rate of jobs to the system

f_i = flow of jobs per unit time through computer i

f_{pj} = flow of jobs (programs) per unit time through communication channel j.

---

†The formulation for $\mu_p \neq \mu_r$ is given by Equation 3.10.

f_{rj} = flow of jobs (results) per unit time through communication channel j.

The expressions for the expected times to pass through each of the arcs in the logical flow network are the expected times to pass through M/M/1 queues with the appropriate mean service times and input rates.

There are two types of constraints that go with the expected job time objective function. There are flow requirements which specify a flow of λ_i jobs from node i to i' and there are capacity constriants which require that $f_i < \ell R_i$ (i=1,2,. . . N) and that $f_{pj} + f_{rj} < \mu C_j$ (j=1,2,. . . NC). The capacity constraints serve to bound the region of feasible flows through the network with boundaries that represent infinite values of the objective function. The flow requirements and capacity constraints together define a convex feasible region which will be denoted by F. This fact, together with the fact that the objective function is convex because it is the sum of convex functions, allows one to apply the multicommodity flow algorithm developed by Cantor and Gerla [Ref. 4] directly to the problem at hand. This algorithm solves for the optimum flow of jobs (f*) through the logical flow network which minimizes E[T] subject to f ε F. The optimal flow solution determines the sets f_i (i=1,2,. . . N) and f_{pj} and f_{rj} (j=1,2,. . . NC) which, together with the inputs λ_i (i 1,2,. . . N) determine the optimal statistical load sharing policy and the system expected job time.

Carrying out the solution of the optimization problem described above requires a computer implementation of the Cantor and Gerla algorithm. A particularly efficient implementation of this algorithm has been developed by Defenderfer [Ref. 6]. The optimization algorithm is based on the efficient generation of extremal flows† of the region F and the subsequent optimization over the region described by these extremal flows. The algorithm does this as follows. One starts with an initial set of extremal flows $(\phi_1, \phi_2, \ldots \phi_j)$ called a basis. Using this basis, a flow f is found which is a convex combination of the basis elements, i.e. $f = \sum_{i=1}^{j} \alpha_i \phi_i$ where $\alpha_i \geq 0$ and $\sum_{i=1}^{j} \alpha_i = 1$. In particular, one finds the f that minimizes $E[T]\left(\sum_{i=1}^{j} \alpha_i \phi_i\right)$ s.t. $\sum_{i=1}^{j} \alpha_i = 1$ and $\alpha_i \geq 0$. This flow will be denoted f*, called the current estimate of the solution to the problem minimize E[T](f) subject to $f \in F$. This flow may only be an estimate because not every element of F can be expressed in terms of the current basis elements.

One now tests if f* is the solution to the main problem minimize E[T](f) s.t. $f \in F$ by checking if there exists an $f \in F$ such that

$$\text{Cost}(f) = \left\langle \nabla E[T](f^*), (f - f^*) \right\rangle < 0$$

where $\langle , \rangle$ is the usual inner product in Euclidian space and ∇ denotes the gradient. If no such f exists, f* solves the main problem. This

---

†An extremal flow is an $f \in F$ which cannot be expressed as a convex combination of any other flows in F.

check is performed by minimizing Cost(f) subject to f ε F. Solving this minimization gives a new extremal flow ϕ_{j+1}. If $\text{Cost}(\phi_{j+1}) \not< 0$, then f* solves the main problem and the algorithm stops. If not, ϕ_{j+1} is added to the set of extremal flows and the algorithm repeats the initial minimization over the new basis. Further details of the algorithm, such as how to find an initial basis point and how to decide when to terminate if an optimal solution has not yet been found, are given in Reference 6.

When solving a statistical load sharing problem, if it is true that in the optimal solution any given communication channel carries either only programs or results, one can write the expected time to pass through a communication channel as

$$E[T]_{comm} = \frac{\left(\frac{f_r}{\mu_r} + \frac{f_p}{\mu_p}\right)(f_r + f_p)}{C - (f_r + f_p)} \qquad (3.9)$$

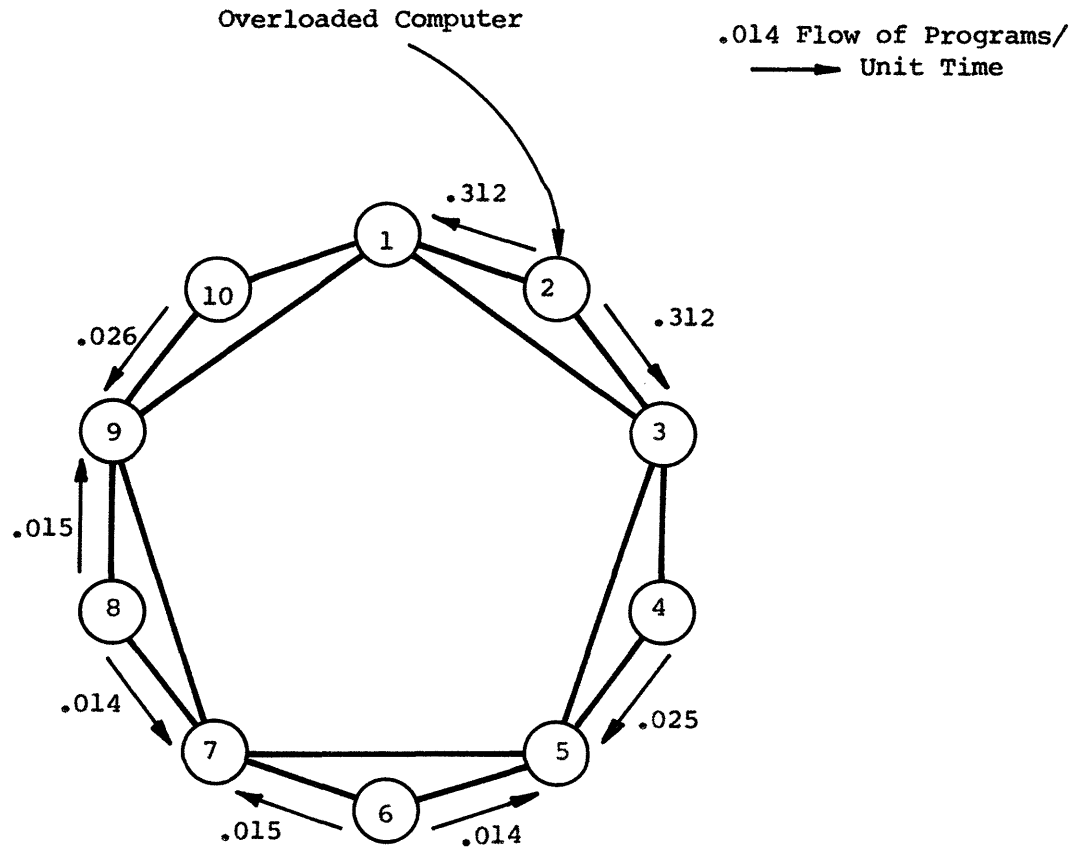
since f_p and f_r will not both be nonzero. In this way one can accomodate problems in which the average message lengths for programs and results are different. When this is done, the general formulation for statistical load sharing becomes

$$\text{Minimize } E[T](f) = \frac{1}{\lambda_T}\left[\sum_{i=1}^{N} \frac{f_i}{\ell R - f_i} + \sum_{j=1}^{NC} \frac{\frac{f_{rj}}{\mu_r} + \frac{f_{pj}}{\mu_p}}{C_j - (f_{rj} + f_{pj})}\right]$$

$$\text{s.t.} \quad f \in F \qquad (3.10)$$

In the previous section, the characteristics of threshold and asymptotic load sharing policies were shown to exist in specific symmetric examples. These characteristics are found in general statistical load sharing problems as well. Figures 3.13 and 3.14 show two load sharing operating points for a ten computer network. In this network, the odd numbered computers have a capacity such that $\frac{1}{\ell R} = 0.5$, while the even numbered computers have a capacity such that $\frac{1}{\ell R} = 2.0$. All communication links are full duplex and the channel capacity is such that $\frac{1}{\mu C} = 2.0$. In both figures the load imbalance is the same, i.e. computer 2 is the overloaded computer. The difference between the two cases is that in Figure 3.13 the total network load is greater. In fact, in Figure 3.13 the network is near saturation and one can clearly see the asymptotic nature of the solution. The load is fairly evenly distributed among computers of equal capacity even though they are at varying distances from the overloaded computer.

Figure 3.14 shows the network in a lightly load situation in which not all of the load sharing flows have reached threshold. Note that this means that there is a "radius" over which load sharing occurs. In large networks, one may want to confine the load sharing to small regions and this example shows that in some cases this may yield an optimal solution.



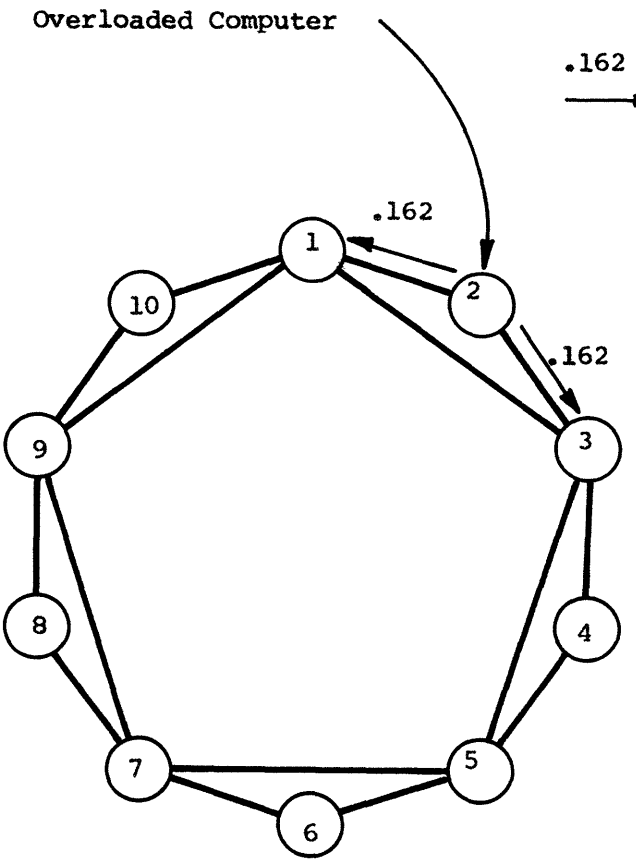
Initial Load Distribution (Jobs/Unit Time)

λ_1	λ_2	λ_3	λ_4	λ_5	λ_6	λ_7	λ_8	λ_9	λ_{10}
1.0	1.0	1.0	0.25	1.0	0.25	1.0	0.25	1.0	0.25

Final Load Distribution (Jobs/Unit Time)

Γ_1	Γ_2	Γ_3	Γ_4	Γ_5	Γ_6	Γ_7	Γ_8	Γ_9	Γ_{10}
1.312	.376	1.312	.225	1.039	.221	1.029	.221	1.041	.224

Figure 3.13 A Ten Computer Load Sharing Example
(λ_T = 7jobs/unit time)



Initial Load Distribution (Jobs/Unit Time)

λ_1	λ_2	λ_3	λ_4	λ_5	λ_6	λ_7	λ_8	λ_9	λ_{10}
.6	.6	.6	.15	.6	.15	.6	.15	.6	.15

Final Load Distribution (Jobs/Unit Time)

Γ_1	Γ_2	Γ_3	Γ_4	Γ_5	Γ_6	Γ_7	Γ_8	Γ_9	Γ_{10}
.762	.277	.762	.15	.6	.15	.6	.15	.6	.15

Figure 3.14 A Ten Computer Load Sharing Example.
(λ_T = 4.2 jobs/unit time)

3.4 A Heuristic Load Sharing Algorithm For Ring Networks

It has been shown that the characteristics of threshold and asymptotic load sharing policies are generally found in statistical load sharing problems. This section uses these characteristics to develop a simple load sharing algorithm for symmetric ring networks (symmetric in the sense that all computers have equal capacity).

The basic idea behind the algorithm is to first test if neighboring computers are above or below the threshold of load sharing. If they are above threshold, the load they are servicing is distributed evenly between them. This is done since even division of load is the asymptotic load sharing policy for equal capacity computers.

Figure 3.15 gives a flow chart for the algorithm. The list of all possible load sharing pairs of computers enables one to keep track of the load sharing decisions already made as the algorithm proceeds. In this way one can avoid inconsistencies such as load sharing both ways between two computers. The equation used to determine if two computers are above or below threshold is a straightforward modification of Equation 3.6. Clearly, if $\lambda_1 > \ell R$ load sharing action must be taken if possible.

As an example of the use of the algorithm, consider the network shown in Figure 3.16. In this network, computers 1 and 2 are overloaded and computers 3 and 4 are underloaded. Figure 3.17 shows the system expected job time for the example when 1) no load sharing is used, 2) the heuristic algorithm is used, and 3) when the Cantor and Gerla algorithm is used. It can be seen that while not optimal, the heuristic algorithm works well. This is particularly significant since the algorithm is so simple. While the algorithm is simple for symmetric ring

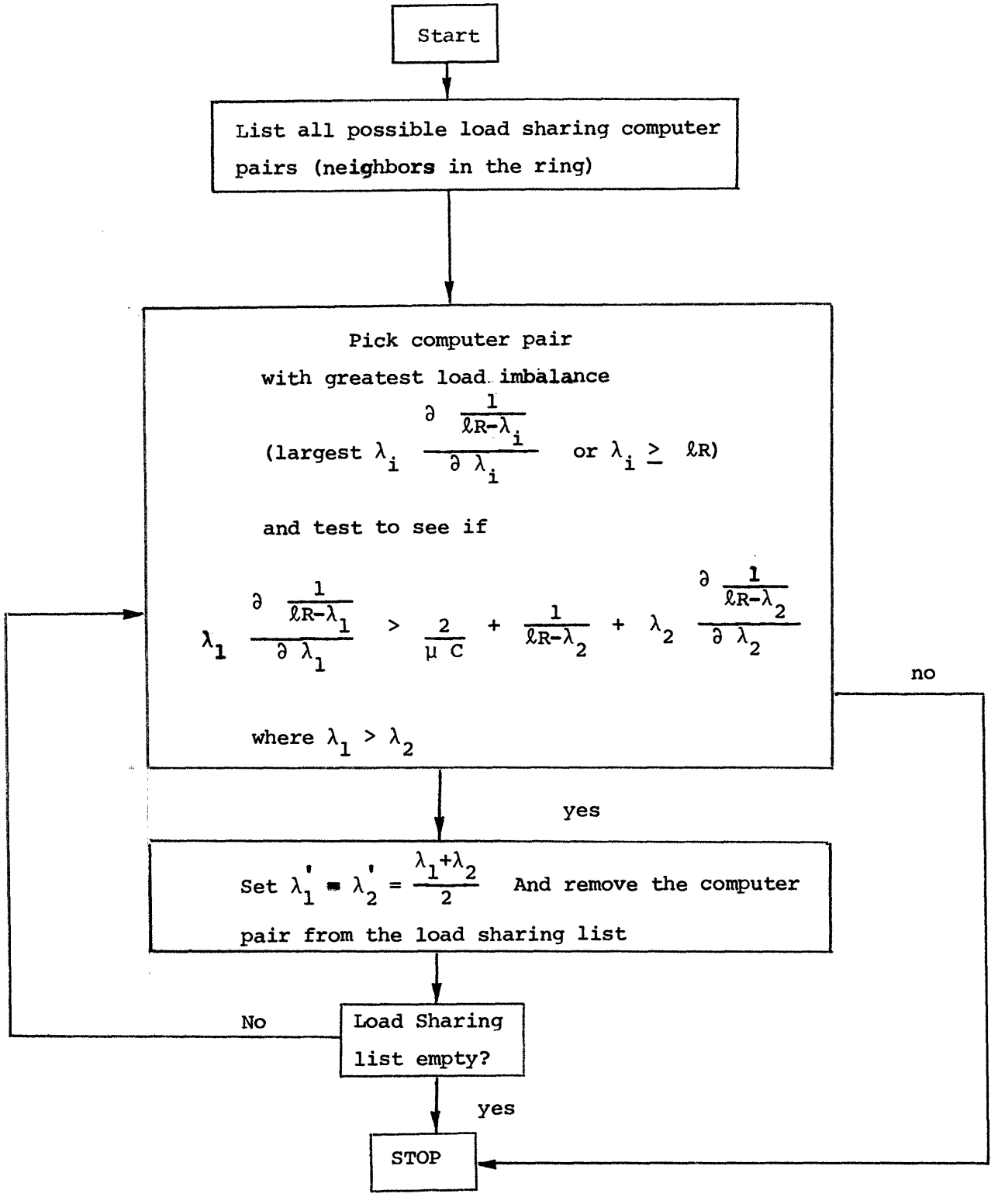
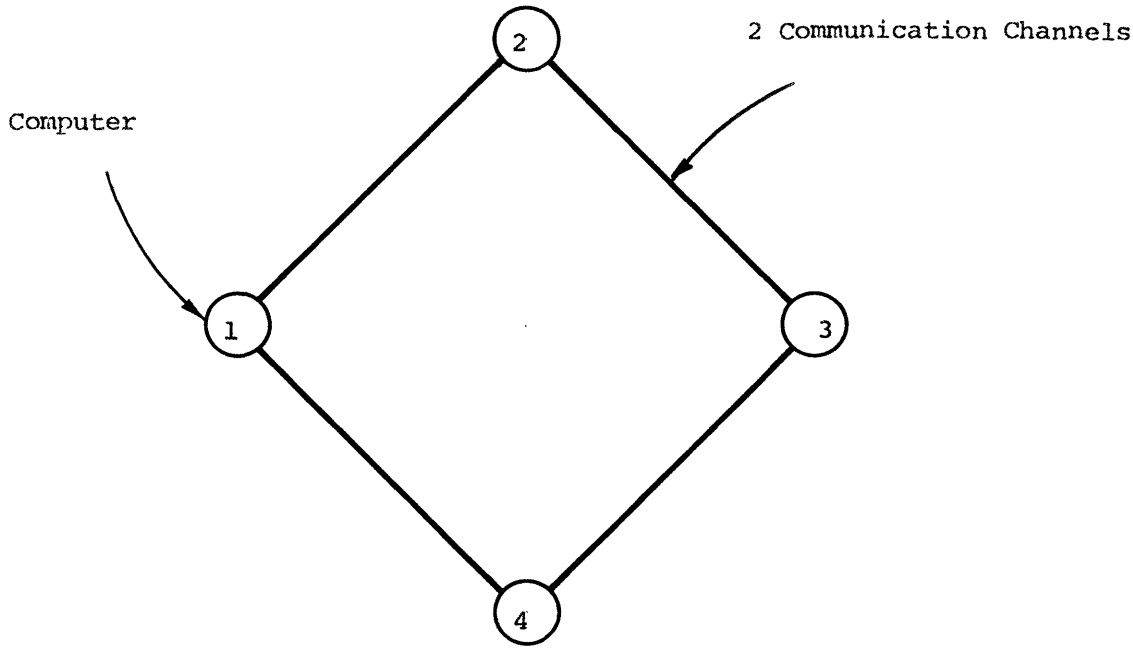


Figure 3.15

Flow Chart of a Heuristic Load Sharing Algorithm for Ring Networks



$1/\ell R = 0.5$ For all computers

$1/\mu C = 0.5$ For all communication channels

Load Imbalance $\lambda_1 : \lambda_2 : \lambda_3 : \lambda_4 = 4:2:1:1$

Figure 3.16 A Four Computer Load Sharing Example

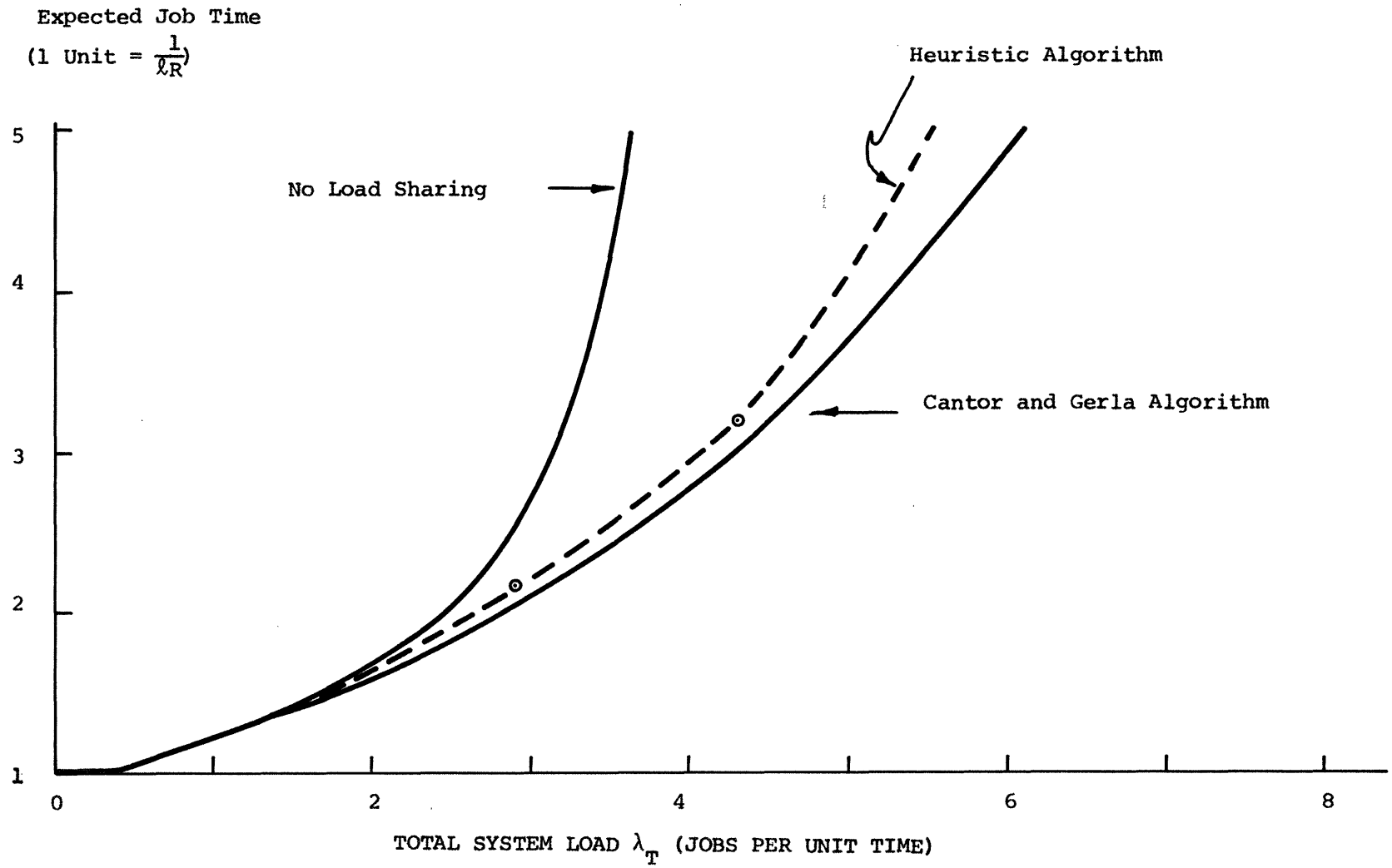


Figure 3.17 Expected Job Time For Four Computer Example Shown In Figure 3.16

networks, however, it is not trivial to extend the underlying ideas to arbitrary networks. This is because the order in which one tests thresholds and makes load sharing decisions is critical if one wants to avoid inconsistencies as mentioned earlier.

3.5 The Effects of Failure in a Load Sharing System

The reliability of a distributed computer system can be improved by load sharing capabilities because load sharing can allow the system to continue to operate at reduced capacity in the event that some of the distributed computers fail. This section considers what happens to system performance when such events occur. This section also shows how system performance can be degraded by the failure of communication links used for load sharing.

Systems in Which Only Computers Fail

In a distributed computer system in which computers are subject to failure, but the facilities for sending jobs to other computers for processing are perfectly reliable, jobs can always be sent to other computers in the system as long as not all of them have failed. Since computer failure in this type of system produces a special case of load imbalance, it is appropriate to consider the performance of statistical load sharing in this situation.

As an example consider a ten computer system which has a fully connected communication network with $1/\mu C = 1/2\ell R$. When a computer fails

in this system, the jobs submitted at the down computer are all sent to be processed elsewhere. If the system load was balanced before the failure, the best policy is to distribute the jobs from the down computer evenly among the computers which are still functioning. Figure 3.18 shows the performance of the ten computer example when this is the case.

It can be seen that statistical load sharing allows the system to operate at a reduced capacity and performance when computers fail in the system. In the example considered here the system can operate up to $\lambda_T = N_\omega \ell R$, where N_ω is the number of computers working in the system. This is because the communication facilities at each computer are sufficient to service all jobs that arrive at down computers. If this is not the case, the system will saturate at $\lambda_T < N_\omega \ell R$. This issue of the saturation of the communication facilities before the saturation of all computer facilities is the same as discussed in Section 3.2.

It is important to note that there is a reduction in system performance as well as system capacity when computers fail. This is because of the communication delay incurred by jobs submitted at down computers and because the computers still operating are now more heavily loaded. If expected job time is critical for the system under consideration, the system may be considered inoperable even when there is still enough computer capacity to process all jobs submitted to the system.

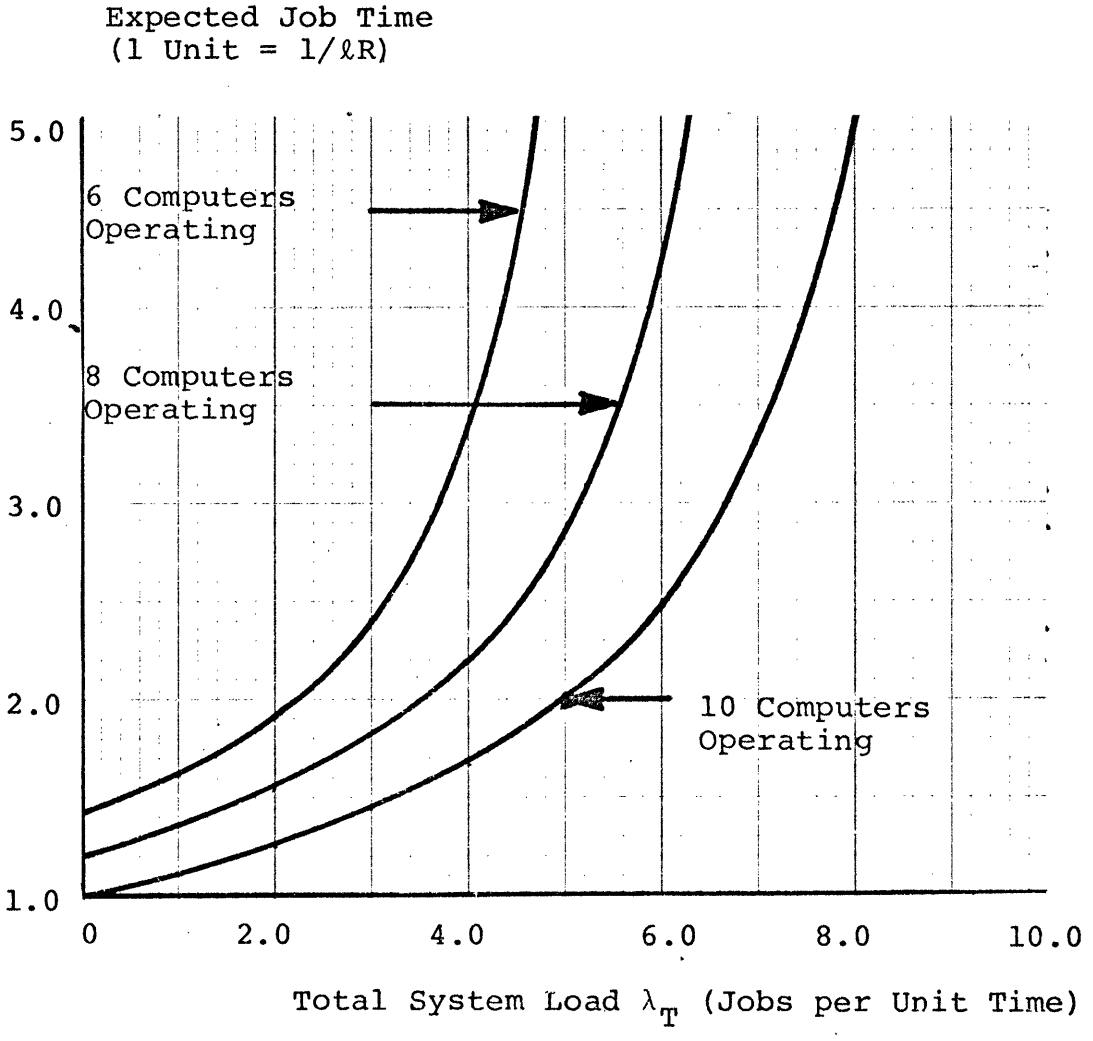


Figure 3.18 Expected Job Time in a Ten Computer System with Computer Failure

The Effects of Communication Channel Failure

The performance of a load sharing system can be degraded by communication channel failure, as well as by computer failure. If the channel which fails carries a significant amount of load sharing traffic, this failure can substantially increase the system expected job time. As an example of this, consider the network shown in Figure 3.16 and assume that the communication channels between computers 1 and 4 fail. The system can continue to operate with this failure, but the system expected job time is increased as shown in Figure 3.19.

In communication networks that are not perfectly reliable, there is a finite probability that not every computer node will have access via communications to every other computer node. This issue related to the reliability of the communication subsystem must be taken into consideration when analyzing the failure characteristics of computer-communication networks. A good survey of the literature on the topic of reliability of the communication subsystem is given by Wilkov [Ref. 35].

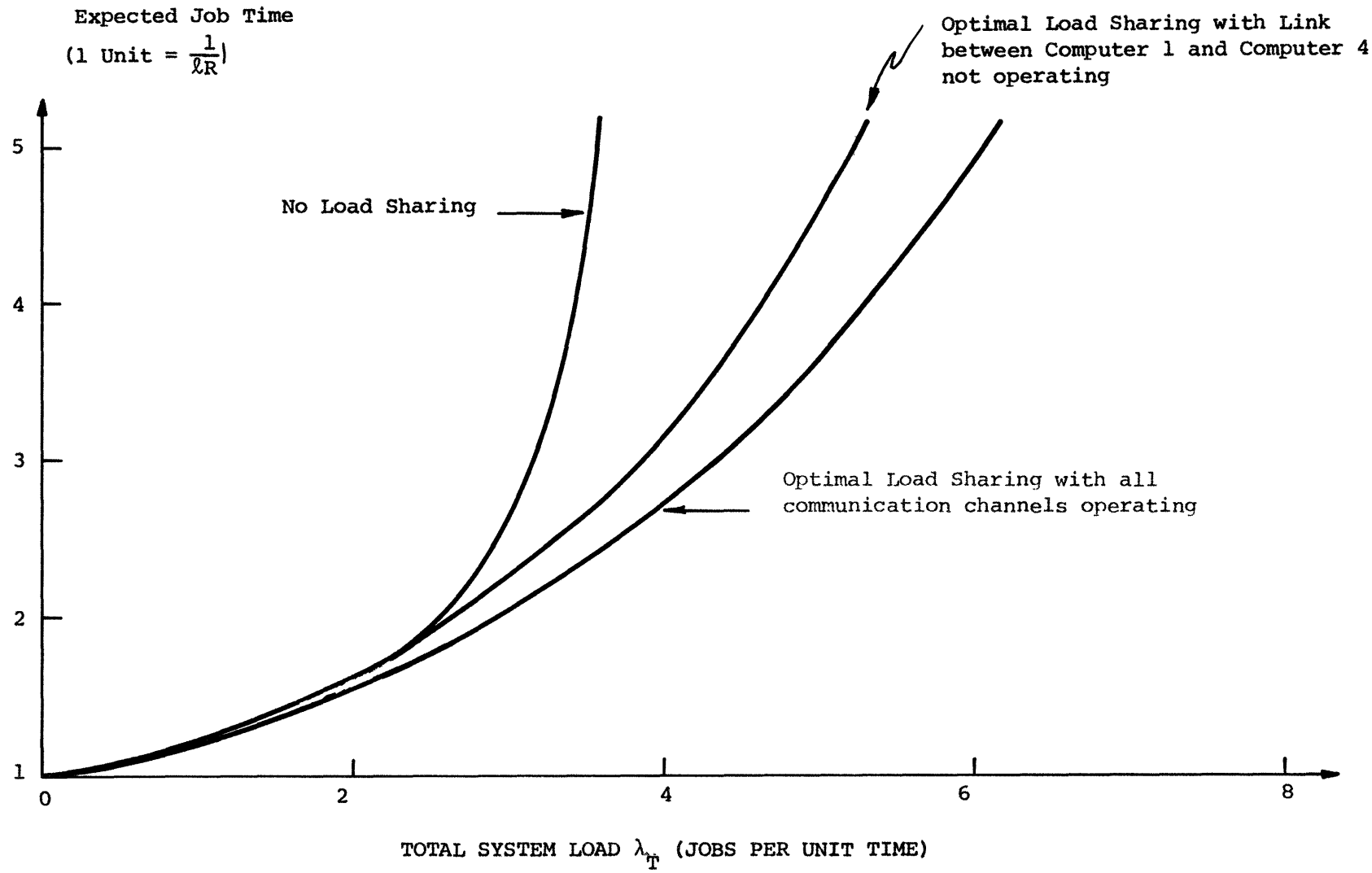


Figure 3.19 Expected Job Time in a Four Computer System With Communication Failure. The network is shown in Figure 3.16.

CHAPTER IV DYNAMIC LOAD SHARING

4.1 Dynamic Load Sharing Using a High Capacity Communication Network

In Chapter 2 it was shown that there is a region of load sharing operation, called the dynamic load sharing region, where one achieves improvements in expected job time beyond those attainable by balancing average loads such as was done with statistical load sharing. In order to achieve such gains it is necessary to use a load sharing technique that assigns jobs to computers on the basis of which computer is the most desirable to use at the time of assignment. This section investigates one such technique which can achieve dynamic gains, starting with a load balanced system, when the communication network being used is a fully connected network of high capacity.

Description of the Dynamic Load Sharing Technique

The dynamic load sharing technique considered here operates with a global controller that uses an instantaneous communication network for control purposes which is separate from the communication network used for load sharing. The controller assigns all incoming jobs to computers on a global first-come-first-served basis. If a job arrives at a time when the computer to which it was submitted is busy, it is immediately assigned to the first available computer according to a preference list. If all computers are busy, the job is queued at the

computer at which it was submitted and is assigned to the first computer that becomes available.

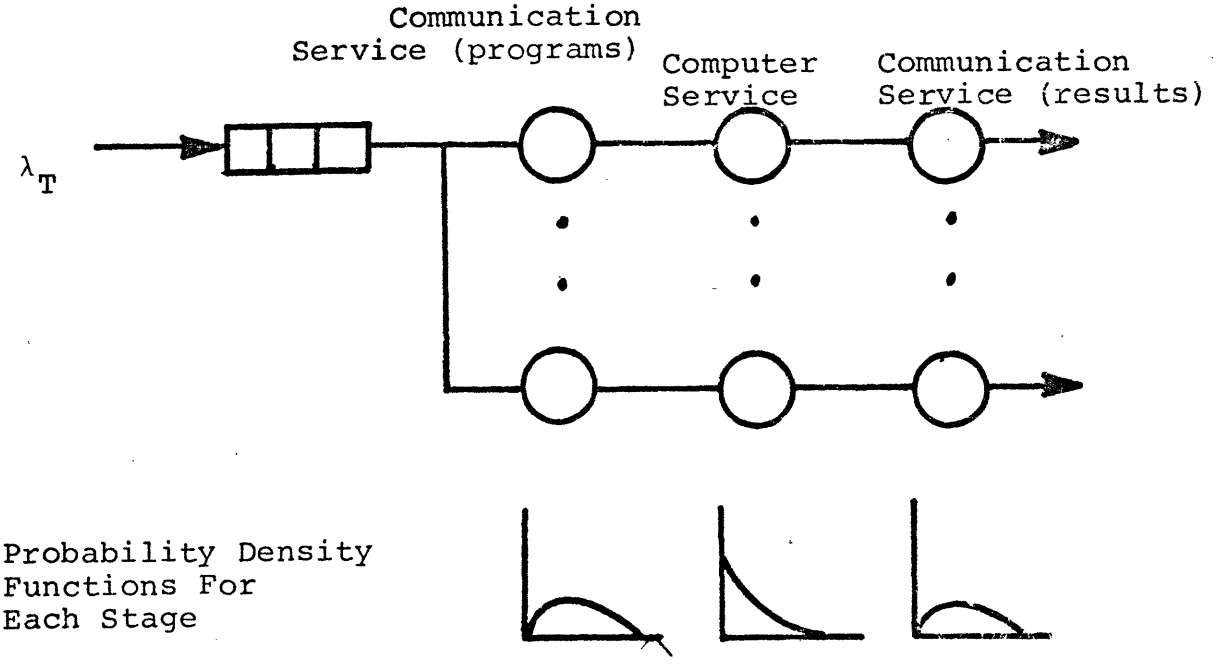
When a job is assigned to a computer other than the one at which it was submitted, the computer to which it is assigned is reserved for it during the time the program is sent to that computer as well as during the time the results are being returned to the computer of origin. This assures that once a job assignment is made, the job will find the computer available when it arrives to be processed.

The performance of this dynamic load sharing technique will now be analyzed by first considering a simple approximation to its performance. This approximation will then be improved by an approximation developed by Larson [Ref. 23] of the hypercube queueing model which describes queueing systems in which servers and customers are identified by spatial locations.

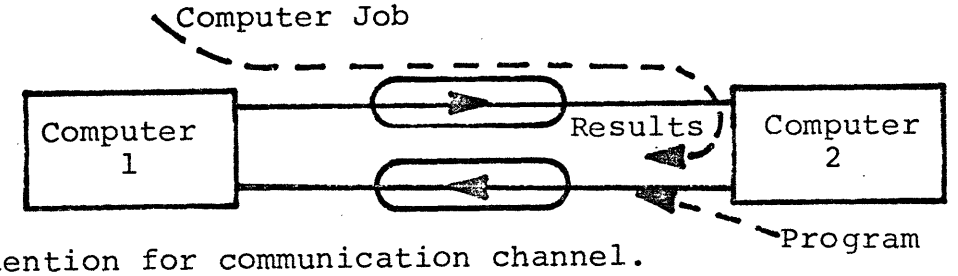
A First Approximation To Dynamic Load Sharing Performance

The dynamic load sharing technique under consideration operates in a manner that is very similar to a multiserver queue. The difference is that in the load sharing system, both arriving jobs and computers are distinguishable as to their spatial location. This means that there is a preferred computer for each job that arrives. It also means that some jobs must undergo a communication delay while others do not.

One way to develop a first approximation of the performance of this dynamic load sharing technique is to consider the performance of a particular multiserver queue that must clearly perform worse than the load sharing system. Such a multiserver is shown in Figure 4.1a. In this multiserver job assignments are made dynamically as in the load sharing system, but it is assumed that every job must undergo a communication delay, whether or not it is processed by the computer at which it was submitted. Clearly, the performance of this multiserver must be worse than that of the actual load sharing system. In order to analyze this multiserver bounding model, one must first determine the service time distributions for each of the computer and communication stages in the queue. The computation time is of course distributed as a negative exponential with mean $1/\ell R$ since once a job is assigned to a computer, it is assured that the computer will be available when the job arrives to be processed. The time to pass through a communication channel, however, involves both waiting time and transmission time. Because the system under consideration has a fully connected communication network, there is only one situation in which queueing occurs in a communication channel. This situation is depicted in Figure 4.1b. A job submitted to Computer 1 was assigned to Computer 2 because Computer 1 was busy at the time of assignment. Before Computer 2 finishes processing this job, Computer 1 becomes available and a job arrives at Computer 2 which is assigned to Computer 1. This means that a computer program is being



a) M/G/N Model.



b) Contention for communication channel.

Figure 4.1 First Approximation Model for Dynamic Load Sharing

transmitted to Computer 1. Meanwhile, Computer 2 finishes the job it was processing and wants to send the results back to Computer 1, but the communication channel is busy. The situation could also have occurred in the reverse order, the results being sent before the program and the program therefore having to wait to use the communication channel. This contention between one program message and one result message is the only queueing in the communication channel that occurs when a fully connected communication network is used with the dynamic load sharing technique presented here.† In consequence, one way to assure that the performance of the first approximation model is indeed worse than that of the actual system is to assume that every message must wait for one other message. The distribution of the time to pass through a communication channel is then a second order Erlang distribution of mean $2/\mu C$ (the convolution of two negative exponentials of mean $1/\mu C$). This is shown in Figure 4.1a.

The first approximation model for dynamic load sharing is now a well defined multiserver queue with a general service time distribution (M/G/N queue). The general service time probability density function of this queue is the convolution of the density functions of each of the computer and communication stages. Because analytic results do not exist for an M/G/N queue, it is necessary to approximate its performance by the performance of a multiserver queue for which results

---

†The queueing referred to here is queueing <u>after</u> a job has been assigned to a computer.

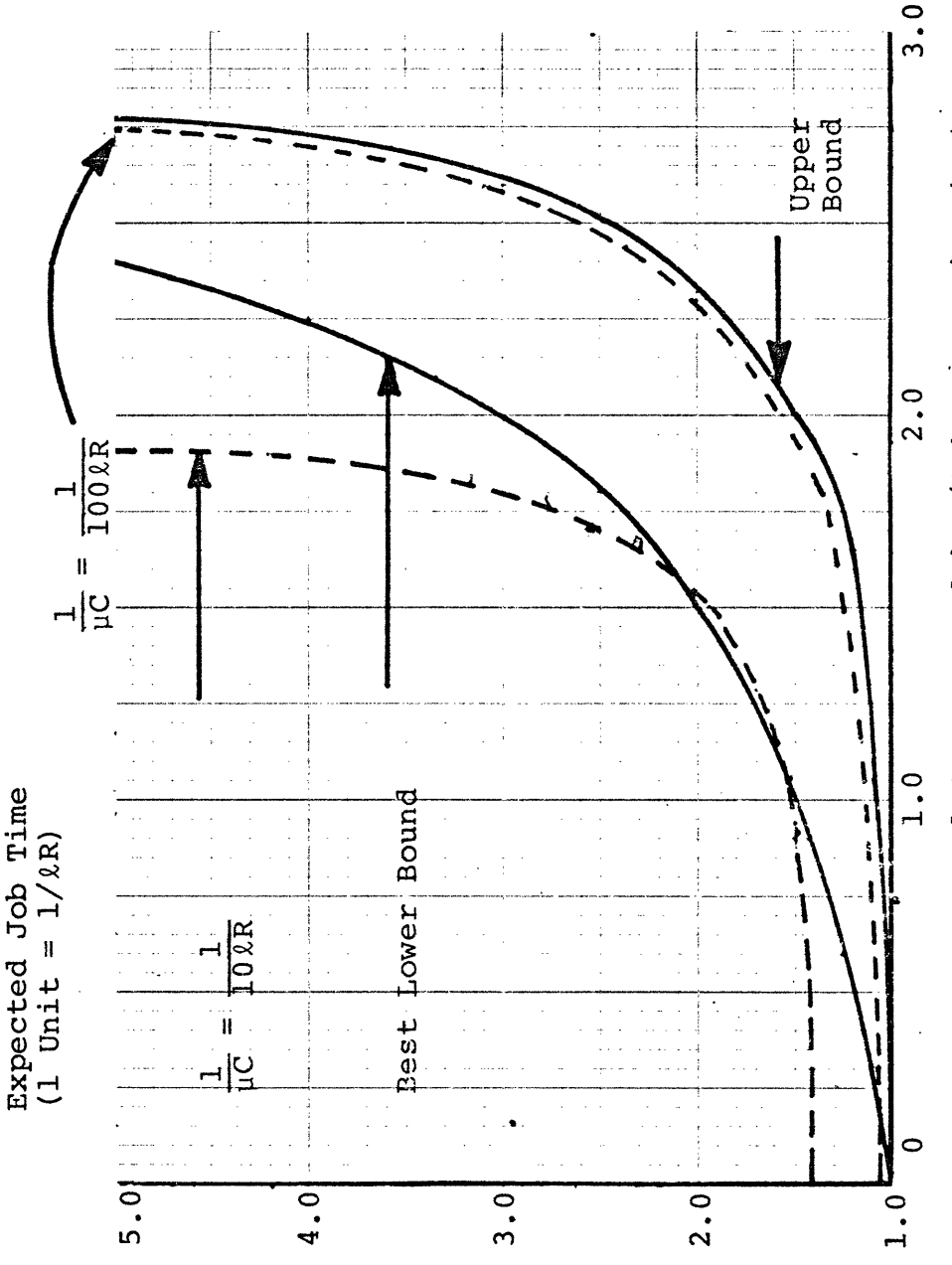


Figure 4.2 First Approximation of Dynamic Load Sharing Performance in a Three Computer System.

are available. The multiserver queue which will be used is a queue with Poisson input and a second order Erlang service time distribution ($M/E_2/N$ queue).[†] Using an $M/E_2/N$ queue, with the same mean service time as the M/G/N queue it is approximating, it is now possible to estimate the performance of the dynamic load sharing technique under consideration. Figure 4.2 shows a graph of this estimate for two different mean communication channel times in a three computer system. Also shown are the upper bound for dynamic load sharing and the performance of a load balanced system of independent computers. The region between these two curves represents operation that is achieving dynamic load sharing gains.

It can be seen that both the computer-communication networks considered achieve dynamic load sharing gains for some values of total system load λ_T. The network with a mean communication channel time $1/\mu C = 1/100\ell R$, does so over a wide range of λ_T and it closely approaches the upper bound, as one would expect for a very high capacity communication network. The network with $1/\mu C = 1/10\ell R$, however, achieves only very small dyanamic gains over a very small range of λ_T. The reason for this is that the first approximation model assumes that every job must undergo a communication delay that involves queueing and transmission time for both programs and results. As a result, for a network

---

[†]The $M/E_2/N$ queue is discussed in Appendix A.

with $1/\mu C = 1/10\ell R$, the mean service time for the corresponding approximation model is $1/\ell R + 4/\mu C = 1.4/\ell R$. This gives an expected job time of $1.4/\ell R$ at $\lambda_T = 0$ and a system pole at $\lambda_T = N\ell R/1.4$, which means that only minimal dynamic load sharing benefits are obtained.

As stated above, the reason that the first approximation model for a network with $1/\mu C = 1/10\ell R$ does not achieve signficant dynamic load sharing gains is that it is assumed that every job incurs a communication delay. In actual system operation this is obviously not true, since some jobs are processed by the computer at which they were submitted. The first approximation of dyanamic load sharing performance will now be refined by using an approximation to the hypercube queueing model to determine the probability of a job being processed at a computer other than the one at which it was submitted. The probability of not sending a job elsewhere to be processed, $1 - P_r$ (send), will then be used to modify the first approximation model as shown in Figure 4.3. The probability of not sending a job has been included as an impulse at the origin of the density function of the communication service time, representing the fact that if a job is not sent, it incurs no communication delay.

Determining the Probability of Sending a Job by an Approximation to the Hypercube Queueing Model

The hypercube queueing model is a multiserver queue which has identifiable servers and customers. It has been used mainly to analyze the

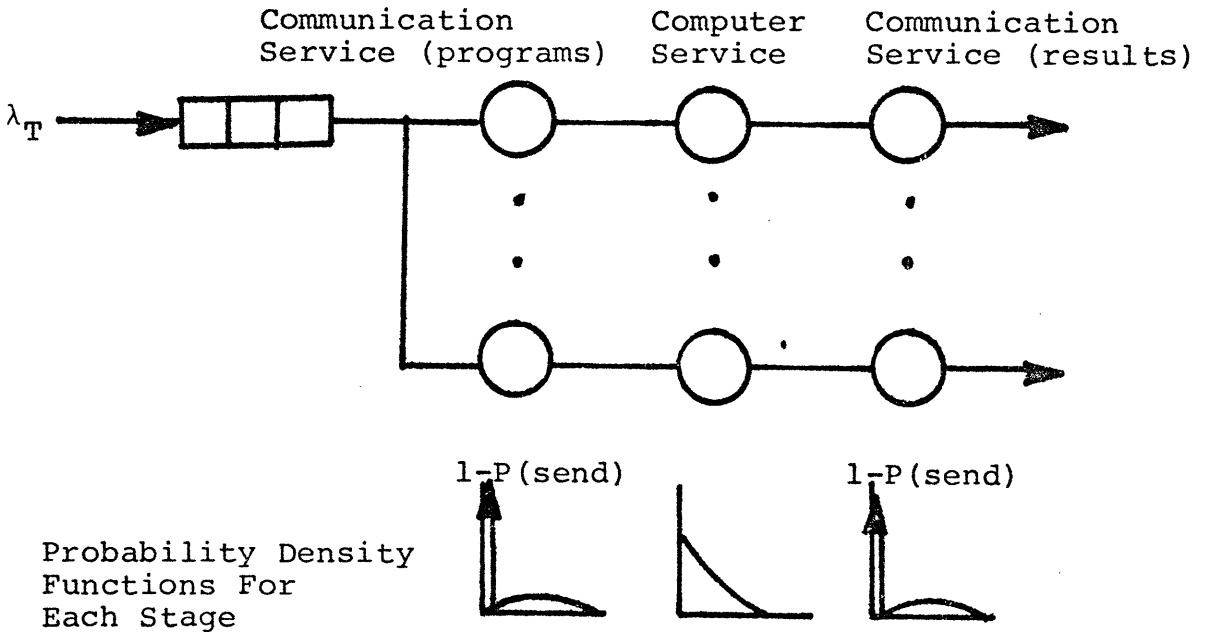


Figure 4.3 Hypercube Approximation Model for Dynamic Load Sharing.

operation of urban emergency service systems in which police cars or other emergency vehicles are the servers and calls for assistance represent customers. Both the servers and customers are identified as to their location. This is analogous to the load sharing system in which jobs and computers are identified as to their location.

The following approximation to the hypercube model is based on the work of Larson [Ref. 23]. It has been modified slightly to fit the load sharing problem. †

In the computer-communication network model, one is interested in determining the probability that a job submitted at Computer i (i=1,2, . . . N) is processed by Computer j (j=1,2,. . . N). Job assignments are made according to the rule that a job is processed by the computer at which it was submitted, if that computer is not busy. If the computer of origin is busy, the job is assigned to the first available computer according to a preference list and if all computers are busy, the job is queued and assigned to the first computer that becomes available.

The probability of sending a job from Computer i to Computer j in a system like this is the probability that in a random sampling without replacement one would find Computer i busy and Computer j free.

---

† The hypercube model assumes that travel time to a customer (equivalent to communication time) is part of the customer service time (equivalent to computation time). The modification that has been made here is that communication time and computation time are considered to be separate. This is accomplished by using the additional communication stages in the multiserver model which makes the system an M/G/N queue. The standard hpyercube approximation model is an M/M/N queue.

In general, in assigning a job, one samples computers in order of preference without replacement until a free one is found. For a system in which each computer is equally loaded,† the probability that a job is sent to the jth preference computer and that the job was not queued is

$$P(B_1 B_2 \cdots B_{j-1} F_j)$$

where

$B_j \equiv$ event that the jth server selected at random is busy.

$F_j \equiv B_j^c \equiv$ event that the jth server selected at random is free.

By using conditional probabilities, one can write

$$P(B_1 B_2 \cdots B_j F_{j+1}) = \sum_{K=0}^{N-1} P(B_1 B_2 \cdots B_j F_{j+1} \mid S_K)\, P(S_K)$$

$$j=1,2,\ldots N-1$$

---

†For a system in which each server is equally loaded, the probabilities $P\{B_1 B_2 \cdots B_{j-1} F_j\}$ can be taken directly as probabilities of sending a job because while the events B_j and F_j refer to servers chosen at random, each server appears to be the same if each is equally loaded. Therefore the probability of finding a particular combination of computers busy and free is the same as finding a random combination busy and free. This is not the case if the servers are unequally loaded. The case of unequal loads is treated by Larson [Ref. 23] and Jarvis [Ref. 14] for emergency service systems in which travel time (equivalent to communication time) is not considered separate from customer service time (equivalent to computation time). The analysis presented here does not extend directly to the case of unequal loads.

where

$S_K \equiv$ event that exactly K servers are busy.

But,

$$P\{B_1B_2 \cdots B_jF_{j+1} \mid S_K\} = P\{F_{j+1} \mid B_1B_2 \cdots B_jS_K\}$$

$$P\{B_j \mid B_1B_2 \cdots B_{j-1}S_K\} \cdot$$

$$\cdots P\{B_1 \mid S_K\} \qquad (4.1)$$

The conditional probabilities on the right hand side of Equation 4.1 are now easily found. For example $P\{B_1|S_K\}$ is the probability that the first server selected at random will be busy, given that K servers in the N server system are busy. Clearly,

$$P\{B_1 \mid S_K\} = K/N$$

Similarly, given that the first selected server is busy and that a total of K servers are busy,

$$P\{B_2 \mid B_1S_K\} = \frac{K - 1}{N - 1}$$

In general,

$$P\{B_i \mid B_1B_2 \cdots B_{i-1}S_K\} = \frac{K - (i - 1)}{N - (i - 1)} \quad i = 1,2 \cdots K + 1$$

$$= 0 \text{ if } i > K + 1$$

Similarly,

$$P\{F_{j+1} \mid B_1B_2 \cdots B_jS_K\} = \frac{N - K}{N - j} \quad j = 0,1, \cdots K$$

$$= 0 \text{ if } j > K$$

Therefore,

$$P\{B_1B_2 \cdots B_jF_{j+1}\} = \sum_{K=j}^{N-1} \frac{K}{N} \frac{K - 1}{N - 1} \cdots \frac{K - (j - 1)}{N - (j - 1)} \frac{N - K}{N - j} P\{S_K\}$$

$$j = 1,2, \cdots N-1$$

Assuming that one can solve for the probabilities $P\{S_K\}$ in the system under consideration, one can easily determine the probabilities $P\{B_1B_2 \cdots B_jF_{j+1}\}$ which give the probability of sending a job to the j+1 preference computer and that the job was not queued before assignment. If the job was queued before assignment then all computers

were busy when it arrived and it is assigned to the first computer which becomes available. The probability of a job being assigned to any particular computer in this case is 1/N in steady state. The total probability of a job being sent to be processed at a computer other than the computer of origin is then

$$
\begin{aligned}
P\{\text{send}\} &= P\{\text{send} \mid \text{no queueing delay}\}\ P\{\text{no queueing delay}\} \\
&\quad + P\{\text{send} \mid \text{queueing delay}\}\ P\{\text{queueing delay}\} \\
&= \sum_{j=2}^{N} P\left\{\text{send to jth preference computer and no queueing delay}\right\} \\
&\quad + \left[\frac{N-1}{N}\right] \cdot P\left\{S_K > N\right\} \\
&= \sum_{j=2}^{N} P\left\{B_1 B_2 \cdots B_{j-1} F_j\right\} \\
&\quad + \left[\frac{N-1}{N}\right] \cdot P\left\{S_K > N\right\} \qquad (4.3)
\end{aligned}
$$

Since the probabilities $P\{B_1 B_2 \cdots B_{j-1} F_j\}$ were derived from an approximation[†] to the hypercube queueing model, they must be normalized

---

[†]The exact solution to the hypercube queueing model is obtained by solving the equations of detailed balance for the continuous time, finite state Markov process which describes the behavior of the system. For systems described by M/M/N multiservers with distinguishable servers, close agreement has been found between the approximation and exact results. [Refs. 23 and 14].

by the condition

$$\sum_{j=2}^{N} P\{B_1 B_2 \cdots B_{j-1} F_j\} + P\{F_1 S_K < N\} = P\{S_K < N\} \qquad (4.4)$$

where

$$P\{F_1 S_K < N\} = \sum_{K=0}^{N-1} P\{F_1 \mid S_K\} P\{S_K\}$$

$$= \sum_{K=0}^{N-1} \frac{N - K}{N} P\{S_K\}$$

A good technique for accomplishing this normalization is to simply scale each of the $P\{B_1 B_2 \cdots B_{j-1} F_j\}$ so that Equation 4.4 is met. When this normalization technique is used one can substitute

$$\sum_{j=2}^{N} P\{B_1 B_2 \cdots B_{j-1} F_j\} = P\{S_K < N\} - \sum_{K=0}^{N-1} P\{F_1 \mid S_K\} P\{S_K\}$$

into Equation 4.3 giving

$$P\{\text{send}\} = P\{S_K < N\} - \sum_{K=0}^{N-1} P\{F_1 \mid S_K\} P\{S_K\}$$

$$+ \left[\frac{N - 1}{N}\right] \cdot P\{S_K > N\} \qquad (4.5)$$

The probability of sending a job can now be easily determined. As stated previously, one can then include the probability of not sending a job, 1 - P{send}, as an impulse at the origin of the density function for the service time of the communication stages as shown in Figure 4.3. This improves the approximation of dynamic load sharing performance by reducing the expected communication time. The improved approximation will now be used to examine several examples.

Examples of Dynamic Load Sharing Performance

The following are examples of the previously described dynamic load sharing technique used in computer-communication networks where the average rates of inputs at all computers are the same. The preference list for dynamic assignments is such that this balance is maintained. The networks considered are all fully connected communication networks so that the previous discussion about queueing in communication channels applies.

Consider first a three computer system in which the mean communication time for one channel is $1/\mu C = 1/10\ell R$. A graph of the probability of sending a job vs. system load for this case is shown in Figure 4.4. Note that near $\lambda_T = 0$ the probability of sending a job is zero because the computer of origin is always available when a job arrives. The system also has a pole, at which point $P\{send\} = \frac{N-1}{N}$. This is because, at system

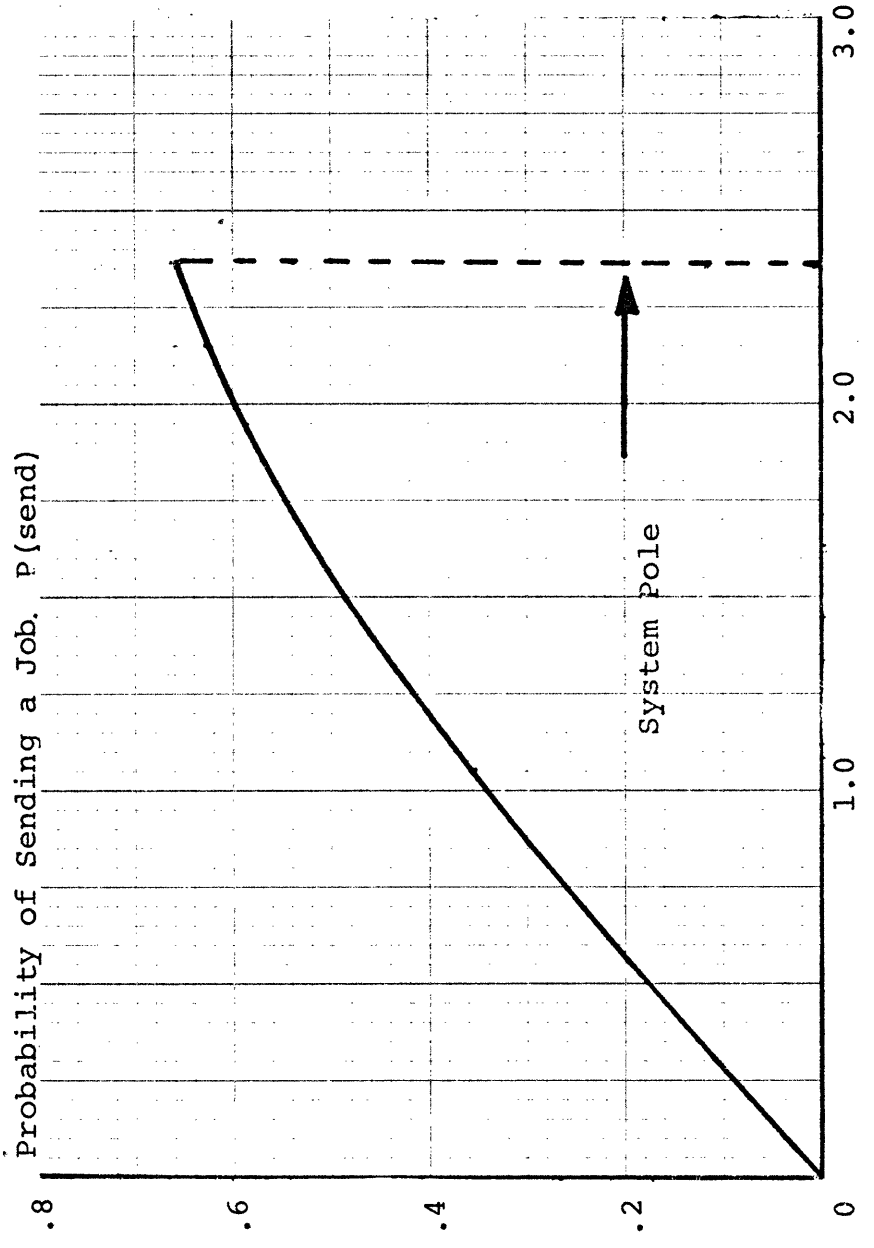


Figure 4.4 Probability of Sending a Job in a Three Computer System

saturation, a job is queued whenever it arrives and when this occurs the probability of sending a job is $(N - 1)/N$. The system pole occurs when the system utilization factor, ρ, equals 1. At this point the mean service time is $1/\ell R + P\{send\}(4/\ell R)$. This means that saturation occurs when $\lambda_T = N\ell R/[1 + .4P\{send\}]$.

A graph of expected job time vs. system load for this three computer case is shown in Figure 4.5. It can be seen that use of the hypercube approximation to determine the probability of sending a job, puts the performance curve for a system with $\frac{1}{\mu C} = \frac{1}{10\ell R}$ well within the dynamic load sharing region. As explained above, the system still has a pole at $\lambda_T < N\ell R$, but for load levels less than $\lambda_T = 2.1$, dynamic gains are clearly indicated.

Examples of five and ten computer systems will now be considered in order to show how our estimates of dynamic load sharing gains vary as a function of system size. Figures 4.6 and 4.7 show the performance curves for five and ten computer systems respectively. As before $\frac{1}{\mu C} = \frac{1}{10\ell R}$. These examples show that as system size increases, better dynamic performance is attained at load levels below the system pole. This is because as the size of the system increases, the probability that a job will be queued before being assigned to a computer decreases.

Figures 4.6 and 4.7 also show a slight shift of the system pole as the number of computers in the system changes. The system pole for dynamic load sharing occurs when

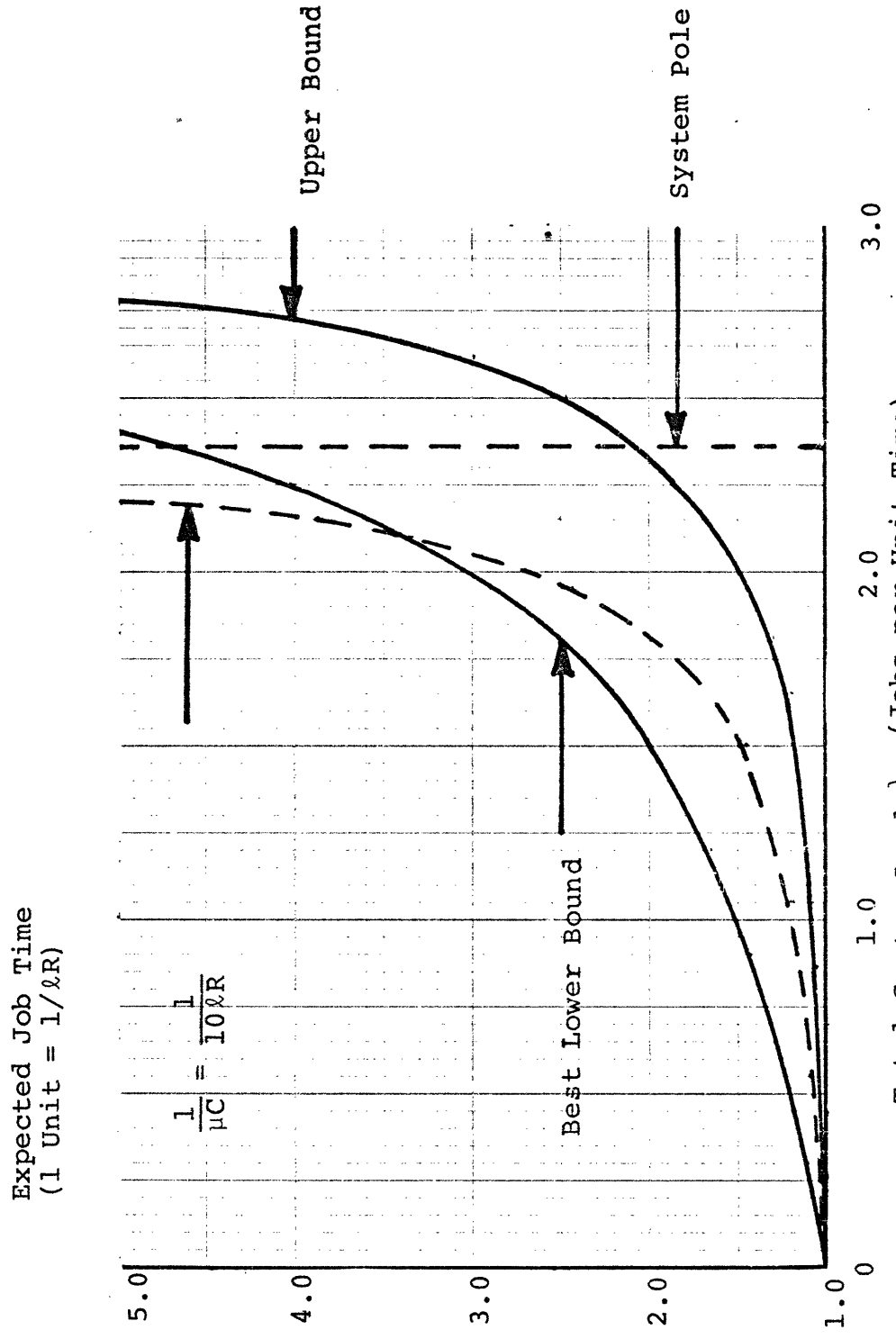


Figure 4.5 Expected Job Time Using Dynamic Load Sharing in a Three Computer System.

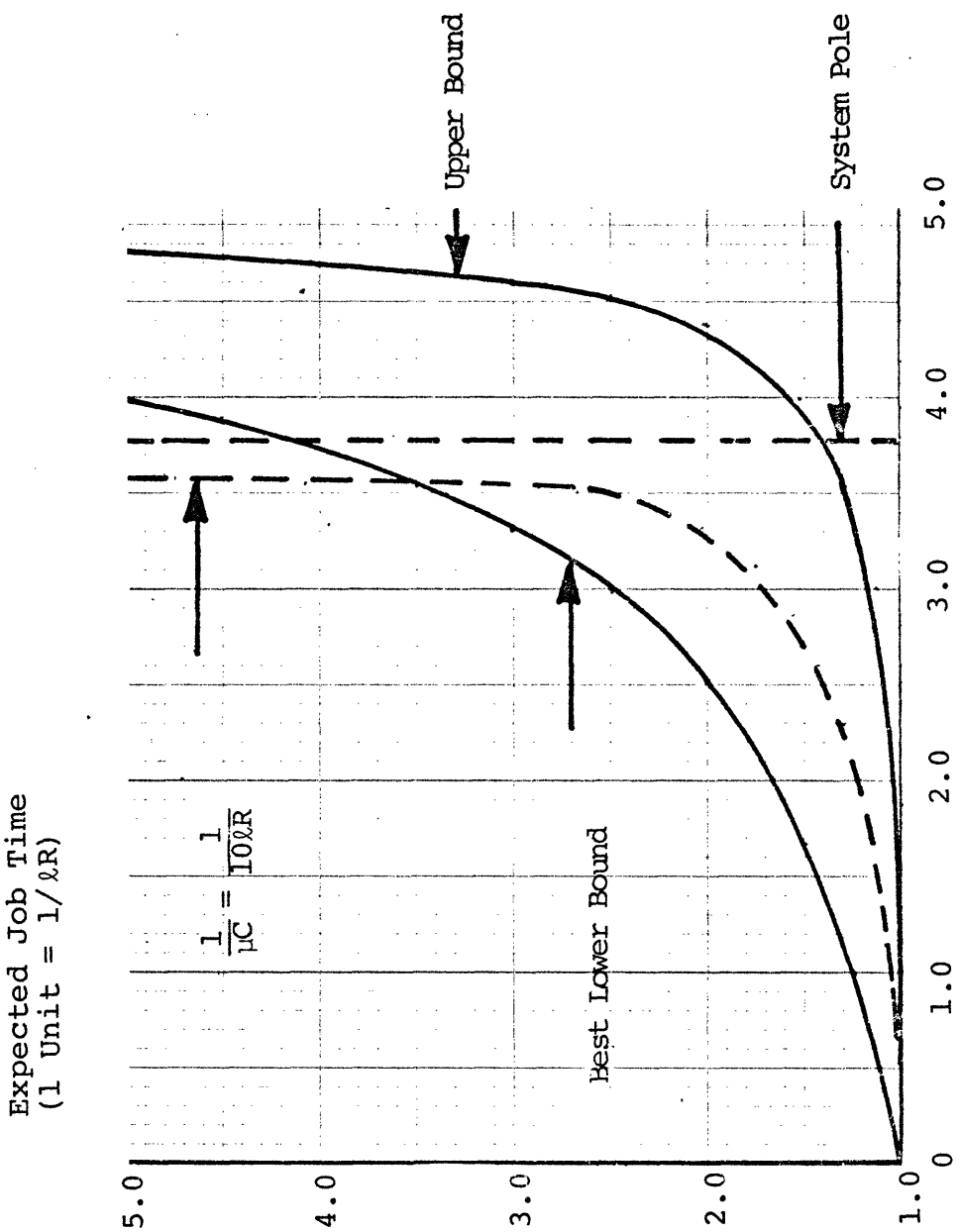


Figure 4.6 Expected Job Time Using Dynamic Load Sharing in a Five Computer System.

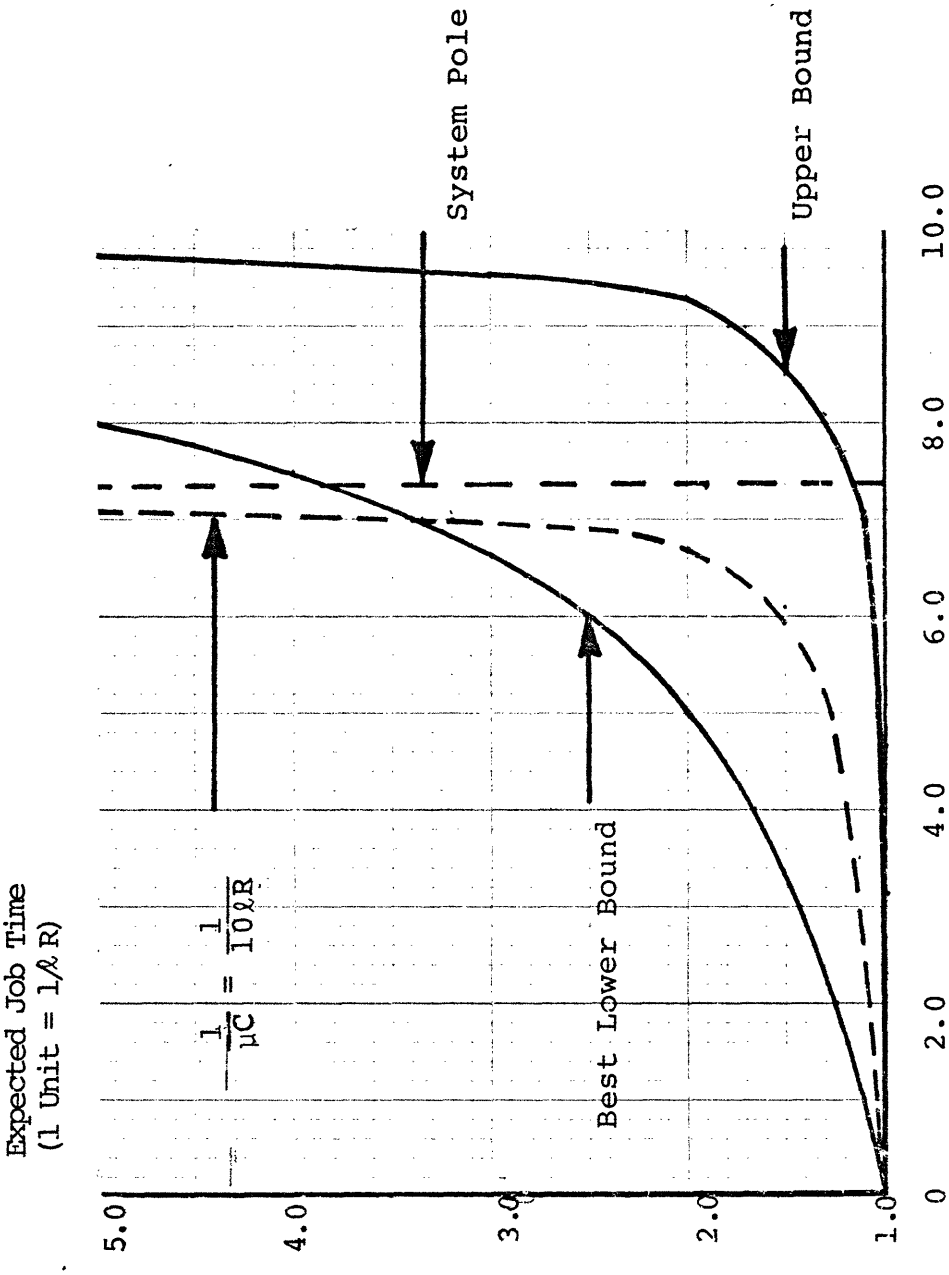


Figure 4.7 Expected Job Time Using Dynamic Load Sharing in a Ten Computer System.

$$\lambda_T \text{ at saturation} = N/[\text{mean service time}] = N/\left[\frac{1}{\ell R} + \frac{4P\{\text{send}\} \text{ at saturation}}{\mu C}\right]$$

$$= N/\left[\frac{1}{\ell R} + \frac{4(N-1)}{N\mu C}\right]$$

Therefore

$$\frac{\lambda_T \text{ at saturation}}{N\ell R} = \frac{1}{\left[1 + \frac{4(N-1)}{N\mu C}\frac{\ell R}{}\right]} \qquad (4.6)$$

Since the term $(N - 1)/N$ varies slowly as a function of system size, the system pole location relative to $N\ell R$ changes slightly as system size changes.

Equation 4.6 can be used to examine the system pole location as a function of mean communication service time. The location of the system pole is determined primarily by the ration $\ell R/\mu C$. For high capacity communication systems, this ratio is small and the system pole is therefore close to $\lambda_T = N R$. As communication capacity decreases, the ratio $\ell R/\mu C$ increases and the system pole moves toward the origin. For this reason, the dynamic load sharing technique described here is effective only with high capaicty communication networks.

The reason that the dynamic load sharing technique presented here requires a high capaicty communication network is that if a job is sent to be processed at a computer other than the one at which it was sub-

mitted, the computer used to process the job is reserved for it during the communication time required to send the job. This is done so that the job is assured of finding the computer available when it arrives to be processed. This works well for a high capacity communication network, but for a low capacity network it means that computers are reserved for large amounts of time during which they provide no service. This moves the system pole towards the origin as communication capacity is decreased (communication delay is increased). Therefore, a dynamic load sharing technique used with a low capacity communication network must eliminate the reservation of computers during communication time, as is discussed in the next section.

4.2 Dynamic Load Sharing Using a Low Capaicty Communication Network

As shown in the previous section, when using a dynamic load sharing technique in a low capacity communication network, one cannot afford to reserve computers during the time jobs are communicated to them. This means that when a job is sent to be processed at another computer in a low capacity system, it may incur a queueing delay at the processing computer in addition to the significant communication delay it incurs. One must therefore consider the policy of queueing a job at the computer of origin when that computer is busy, even if there is another computer that is not busy in the network.

If one considers queueing jobs at busy computers while others are not busy, one is faced with the question of how many jobs to queue before it is better to send a job elsewhere to be processed. Since the hypercube model does not allow this type of operation, one must seek other techniques of analysis. Conceptually, one way in which to analyze a system which allows this type of operation is to model each of the computers and communication channels as queues operating in discrete time with finite length buffers. The system operation is then described by a discrete time, finite state Markov process which could be analyzed for various dynamic load sharing policies. At present, however, it is not feasible to use this analysis because of the extremely large state space required by the problem. The analysis of general dynamic load sharing techniques therefore remains an area open for further research.

CHAPTER V CONCLUSION AND SUGGESTIONS FOR FURTHER RESEARCH

5.1 Conclusion

This study of load sharing in a computer-communication network has shown that load sharing can provide improvements in the expected time to process jobs in a distributed computer system. Upper and lower bounds for this performance criteria were developed and two techniques for load sharing were investigated using queueing models. Specifically, it has been shwon that statistical load sharing can be used to improve expected job time by correcting load imbalances. Most importantly, the correction of these load imbalances allows the system to operate at higher throughput levels than is possible without load sharing. To obtain improvements in expected job time beyond those possible with a simple technique such as statistical load sharing, it is necessary to use a dynamic load sharing technique. One such technique was investigated and shown to give significant dynamic gains if used with a high capacity communication network. It was also shown that load sharing capabilities can improve system reliability by making the system fail soft, at the expense of degraded performance.

Computer-communication networks today are increasing the capabilities of computer systems by providing the means for remote access to time-shared computer facilities, data base sharing and the sharing of unique computer resources. Because of the definite improvements in system expected job time and reliability that load sharing can provide, provisions

for load sharing should be given serious consideration in the design of future computer-communication networks. A study of the performance curves for statistical load sharing and dynamic load sharing shows that it is most important to balance out severe load imbalances. The performance improvement that is gained by simply balances. The performance improvement that is gained by simply balancing the average load is far greater than the additional improvement gained by doing dyanamic job assignment. This indicates that in actual implementations of load sharing, it may be sufficient to make load sharing policy decisions on a periodic basis to balance average loads, rather than to make a decision based on system state for every job. An important result of this study is the identification of the load sharing problem as a multicommodity flow problem. This means that as progress is made in solving the problem of dynamic control of other multicommodity flow problems, such as message routing in a packet switched communications network, the results can be applied to the load sharing problem.

5.2 Suggestions for Further Research

The upper and lower bounds on system performance, that were developed in Chapter 2, provide a frame of reference within which to evaluate load sharing techniques. It would be of interest to examine load sharing techniques other than those presented here, such as the dynamic technique of Roome and Torng [Ref. 31], within this frame of reference.

In order to achieve a better understanding of the benefits of load sharing, it would be of interest to obtain a more complete statistical description of performance than just expect job time. Higher moments of job time distributions, or even better, complete job time distributions would be of value.

Since a computer-communication network is a dynamic system, it is important to understand its transient operation as well as its steady state operation. A transient situation that is of particular interest is the system response to a temporary overload at one of the computers.

As suggested in Chapter 4, another idea for further study is to model each of the computers and communication channels as queues operating in discrete time with finite length buffers and to use a discrete time, finite state Markov process analysis to study general dynamic load sharing techniques. In order to use this approach, one must first find ways to handle the problem of the extremely large state space generated by this model.

Another suggestion for further research is to consider reliability improvements using load sharing techniques in systems where the communication channels are subject to degradation rather than total failure. An example of such degradation is a change in signal to noise ration in a radio channel.

A final suggestion for further research is to investigate dynamic load sharing operation using control schemes that do not assume a global controller using an instantaneous communication system separate from

the communication network used for load sharing. It is quite likely that in actual implementations of load sharing, there would not be a separate global controller and that control information would be sent via the same communication network as the computer programs and results.

APPENDIX A QUEUEING FORMULAS

This appendix gives the basic queueing formulas used in this study. Derivations of the formulas in Sections A.1 and A.2 can be found in standard references on queueing theory such as Cohen [Ref. 5] (M/M/1 queue only), Gross and Harris [Ref. 8], Hillier and Lieberman [Ref. 10] or Saaty [Ref. 33].

A.1 The M/M/1 Queue

The M/M/1 queue is a single server queue with a Poisson arrival process with mean arrival rate λ and a negative exponential service time distribution with mean service time $1/\mu$. The queue has a steady state solution only when the utilization factor, $\rho = \lambda/\mu$, is less than 1. If this is the case, the steady state disbritubion of the number of customers in the system is given by

$$P_K = (1 - \rho)\ \rho^K \qquad K = 0,1,2. \ . \ .$$

where $\rho = \lambda/\mu < 1$

Using this steady state distribution, it can be shown that the distribution of the time to pass through the M/M/1 queue is also exponential with mean

$$E[T] = \frac{1}{\mu - \lambda}$$

A.2 The M/M/N Queue

The M/M/N queue is an N server queue, also with Poisson input (mean arrival rate λ) and exponential service time (mean service time $1/\mu$). The steady state distribution of the number of customers in the system is given by

$$P_K = \begin{cases} \dfrac{(\lambda/\mu)^K}{K!} P_0 & \text{if } 0 \leq K \leq N \\ \dfrac{(\lambda/\mu)^K}{N!\ N^{K-N}} P_0 & \text{if } K > N \end{cases}$$

where

$$P_0 = 1 \Bigg/ \left[\sum_{n=0}^{N-1} \frac{(\lambda/\mu)^n}{n!} + \frac{(\lambda/\mu)^N}{N!} \frac{1}{1 - (\lambda/N\mu)} \right]$$

By solving for the expected number of customers in the system and applying $L = \lambda W$, it follows that the expected time to pass through the system is given by

$$E[T] = \frac{P_0 (\lambda/\mu)^N \rho}{\lambda\ N!\ (1-\rho)^2} + \frac{1}{\mu}$$

where $\rho = \lambda/N\mu$

A.3 The $M/E_k/N$ Queue

The $M/E_k/N$ queue is an N server queue with a k th order Erlang service time distribution. The technique used to analyze this queue is given by Heffer [Ref. 9]. The distribution of the number of customers in the system and the resulting value of the expected time to pass through the system are not given by convenient closed form solutions. Numerical results, however, are available in Hillier and Lo [Ref. 11]. These numerical results were used to perform the calculations in this study.

In order to calculate performance curves for the hypercube approximation model, the sequence of calculations that one follows is

1. Given the value of the total system utilization factor, calculate the probability of sending a job.

2. Using the probability of sending a job and the system utilization factor, calculate the total system load λ_T at which this utilization factor occurs.

3. Calculate the expected job time by using Little's formula $L = \lambda W$ [Ref. 23] where L is the expected number of customers in the system at this system utilization factor and W is the desired job time.

APPENDIX B PROOF OF THE APPLICABILITY OF THE NETWORK OF QUEUES MODEL TO STATISTICAL LOAD SHARING

The analysis of the network of queues model for statistical load sharing was made possible by a theorem by Jackson [Ref. 12] which states that in steady state, the probability distribution of the state of the network can be written in a product form. The terms in the product are the distributions of the state (number of customers) at each queue in the network considered as a separate independent queue with the appropriate input rate. The purpose of this appendix is to show that the statistical load sharing problem meets the requirements of the Jackson theorem.

The requirements of the Jackson theorem are that

1. Customers from outside the system arrive at each queue as a Poisson stream.

2. Once served at queue m, a customer's destination is determined by a random sampling. With probability θ_{KM} he goes (instantaneously) to queue K (k = 1,2, . . . M) and with probability

$$1 - \sum_{k=1}^{M} \theta_{KM}$$

he leaves the system.

3. Each queue has an exponential service time distribution and serves all arriving customers (from inside or outside) in a first-come-first-served manner.†

The third requirement is clearly met by the statistical load sharing problem, since both communication channels and computers are modeled

---

†A first-come-first-served discipline is sufficient, but not necessary. [Ref. 22].

as exponential servers.† The other two requirements, however, need to be examined more closely.

The requirement that customers from outside the system arrive as a Poisson stream is met by assumption at all computer queues which do not send jobs on to be processed elsewhere. At computer queues where some of the jobs are sent elsewhere, customers arrive as a random process which is a random sampling of a Poisson input stream. This is also true of communication queues which are used to forward jobs arriving from outside the system (computer programs). Therefore, in order to show that the first requirement of the Jackson theorem is met, it is necessary to show that a random sampling of a Poisson process yields another Poisson process. Such a proof is given below.

Proof that a Random Sampling of a Poisson Process Yields Another Poisson Process

A Poisson arrival process with rate parameter λ is a renewal process for which the interarrival times are exponentially distributed with mean $1/\lambda$ [Ref. 16]. Therefore, for such a process, the Laplace transform of the interarrival time distribution is

$$L\{f_T(t)\} = \int_{t=-\infty}^{\infty} e^{-st} f_T(t)\,dt = \int_{t=0}^{\infty} e^{-st}\,\lambda e^{-\lambda t}\,dt = \frac{\lambda}{s+\lambda}$$

---

†As noted in Chapter 3, the independence assumption [Ref. 17] for communication channels may be required.

Now consider a random sampling of this process in which arrivals are counted with probability β and not counted with probability $1 - \beta$. Then the Laplace transform of the density function for the time L between successive arrivals which are counted is

$$L\{f_L(\ell)\} = L\{f_{W_1}(w_1)\}\beta + L\{f_{W_2}(w_2)\}\beta(1-\beta) + L\{f_{W_3}(w_3)\}\beta(1-\beta)^2 + \ldots.$$

$$= \sum_{n=1}^{\infty} L\{f_{W_n}(w_n)\}\beta(1-\beta)^{n-1}$$

where W_n is the sum of n interarrival times in the underlying Poisson process.

Since the transform of the density of a sum of statistically independent random variables is the product of the transforms independent random variables is the product of the transforms of each random variable in the sum, the expression becomes

$$L\{f_L(\ell)\} = \sum_{n-1}^{\infty} \left(\frac{\lambda}{s+\lambda}\right)^n \beta(1-\beta)^{n-1} = \frac{\beta\lambda}{s+\beta\lambda}$$

Therefore L is distributed exponentially with mean $1/\beta\lambda$. Since successive interarrival times in the sampled process are also statistically independent, the sampled process is Poisson with rate parameter $\beta\lambda$. Q.E.D.

The final requirement that the statistical load sharing problem must meet is that, once serviced at a queue, the destination of a job is determined by random sampling. This results in a random routing through the network of queues. In the statistical load sharing problem, the routing of a job is random until it has passed through a computer queue. Once it has passed through a computer queue, it must be returned to the computer of origin before it can leave the system. This deterministic routing in the statistical load sharing problem must therefore be shown to still meet the requirements of the Jackson theorem.

Consider a computer queue which services both jobs submitted to it directly (arriving at rate λ_1) and jobs sent from an overloaded computer (arriving at rate λ_2). At the output of this queue, it must be decided whether a job leaves the system (if it was submitted to the computer directly) or if it is to be sent over a specific communication channel (if it came from the overloaded computer). Assume that the decision is made on the basis of a tag which identifies the origin of the job. In order for this decision to meet the requirements of the Jackson theorem, it must produce output streams of customers that appear as if the decision was made by random sampling.

When a random decision rule is used, the output streams from the computer queue are both Poisson. This follows from the fact that the output of an exponential server with Poisson input is Poisson, as has

been shown by Burke [Ref. 3]. This property of exponential servers makes all job streams in the network of queues Poisson when a random decision is used. The statistical load sharing problem must therefore also generate Poisson streams at all points in the network.

As stated before, the routing decision at the output of the computer under consideration is made on the basis of a tag which identifies the origin of the job. The sequence of decisions that are made at the output of the queue are generated by the order in which jobs arrive at the input of the computer becuase all jobs are served in a first-come-first-served manner. In a sequence of routing decisions, the probability that the next job is of origin 1 is the probability that, in the input stream, the job which arrived immediately after the job whose routing has just been determined was of origin 1. This probability is $\lambda_1/(\lambda_1 + \lambda_2)$, independent of all previous outputs because jobs arrive at the input of the computer as independent Poisson streams of rates λ_1 and λ_2. The sequence of decisions made at the output, therefore appears to an observer at that point to be a purely random sequence and the resulting output streams with different destinations are therefore Poisson as required by the Jackson theorem.

Another way to show that the statistical load sharing problem meets the requirements of the Jackson theorem is to apply the idea of a job routing determined by an Nth order Markov chain as has been done by Kobayashi and Reiser [Ref. 22].

LIST OF SYMBOLS

N: number of computers in the system.

λ_T: mean total arrival rate of computer jobs in the system.

λ_i: mean arrival rate of computer jobs at the i th computer.

$1/\ell$: mean number of operations required per computer job.

R_i: rate at which the i th computer performs operations.

$1/\mu p$: mean message length in bits for computer programs.

$1/\mu r$: mean message length in bits for computer results.

C_i: channel capacity in bits per unit time of the i th communication channel.

$E[T]$: system expected job time.

$E[T_i]$: expected time to process a computer job which enters the system at the i th computer.

ρ: utilization factor of a queue.

β: probability of sending a job which arrives at an overloaded computer to a specific underloaded computer using statistical load sharing.

f_i: flow rate of computer jobs through the i th computer.

f_{pi}: flow rate of computer programs through the i th communication channel.

f_{ri}: flow rate of computer results through the i th communication channel.

NC: number of communication channels in the network.

$P\{send\}$: probability of sending a job in a dynamic load sharing system.

$P\{B_i\}$: probability of finding the i th computer sampled in a dynamic load sharing system to be busy.

$P\{F_i\}$: probability of finding the i th computer sampled in a dynamic load sharing system to be free-

$P\{S_K\}$: probability of there being k customers in a dynamic load sharing system.

L: expected number of customers in a queueing system.

W: expected time to pass through a queueing system.

REFERENCES

(1) Bowdon, E.K., Sr., "Modeling and Analysis of a Network of Computers", Ph.D. Thesis, Dept. of Electrical Engr., Univ. of Iowa, 1969.

(2) Bowdon, E.K., Sr., "Dispatching in Network Computers", Proceedings of the Symposium on Computer-Communications Networks and Teletraffic, Brooklyn, N.Y.: Polytechnic Press, 1972.

(3) Burke, P.J., "The Output of a Queueing System", Operations Research, Vol. 4, 1956, pp. 699-704.

(4) Cantor, D.G. and Gerla, M., "Optimal Routing in a Packet-Switched Computer Network", IEEE Transactions on Computers, Vol. 23, No. 10, Oct. 1974, pp. 1062-1068.

(5) Cohen, J.W., The Single Server Queue, Amsterdam: North Holland, 1969.

(6) Defenderfer, J., "Comparative Analysis of Routing Algorithms for Computer Networks", M.I.T., Dept. of Electrical Engr. and Computer Science, (Thesis in preparation).

(7) Fuchs, E. and Jackson, P.E., "Estimates of Distributions of Random Variables for Certain Computer Communications Traffic Models", Communications of the ACM, Vol. 13, No. 12, Dec. 1970, pp. 752-757.

(8) Gross, D. and Harris, C.M., Fundamentals of Queueing Theory, New York: John Wiley and Sons, 1974.

(9) Heffer, J.C., "Steady-State Solution of the $M/E_k/C$ (FIFO) Queueing System", CORS Journal, Vol. 7, 1969, pp. 16-30.

(10) Hillier, F.S. and Lieberman, G.J., Introduction to Operations Research, San Francisco: Holden-Day, Inc., 1974.

(11) Hillier, F.S. and Lo, F.D., "Tables for Multiple-Server Queueing Systems Involving Erlang Distributions", Stanford University TR-149, June 1972.

(12) Jackson, J., "Networks of Waiting Lines", Operations Research, Vol. 5, 1957, pp. 518-521.

(13) Jackson, P.E. and Stubbs, C.D., "A Study of Multiaccess Computer Communications", Spring Joint Computer Conference, AFIPS Conference Proceedings, Vol. 34, 1969, pp. 491-504.

(14) Jarvis, J.P., "Optimization in Stochastic Service Systems with Distinguishable Servers", M.I.T. Operations Research Center TR-19-75, June 1975.

(15) Kahn, R.E., "Resource Sharing Computer Communications Networks", Proceedings of the IEEE, Vol. 60, No. 11, Nov. 1972, pp. 1397-1407.

(16) Karlin, S., A First Course in Stochastic Processes, New York: Academic Press, 1969.

(17) Kleinrock, L., Communication Nets: Stochastic Message Flow and Delay, New York: McGraw-Hill, 1964.

(18) Kleinrock, L., "Models for Computer Networks", Proc. of the International Communications Conference, Univ. of Colorado Boulder, June, 1969, pp. 21-9 to 21-16.

(19) Kleinrock, L.,"Analysis and Simulation Methods in Computer Network Design", Spring Joint Computer Conference, AFIPS Conference Proceedings, Vol. 36, 1970, pp. 569-579.

(20) Kleinrock, L., "Scheduling, Queueing, and Delays in Time-Shared Systems and Computer Networks", Computer-Communication Networks, Abramson, N. and Kuo, F. Eds., Englewood Cliffs, N.J.: Prentice-Hall, Inc. 1973.

(21) Kleinrock, L., "Resource Allocation in Computer Systems and Computer-Communication Networks", Information Processing 74, Proceedings of the IFIP Congress 74, Amsterdam: North Holland, 1974, pp. 11-18.

(22) Kobayashi, H. and Reiser, M., "On Generalization of Job Routing Behaviour in a Queueing Network Model", IBM Research RC 5252, Feb. 1975.

(23) Larson, R.C., "Approximating the Performance of Urban Emergency Service Systems", preprint to appear in Operations Research, Operations Research Center, M.I.T. PP-03-74, Feb. 1975.

(24) Little, J.D.C., "A Proof of the Queueing Formula L = W", Operations Research, Vol. 9, No. 3, May-June 1961, pp. 383-387.

(25) Martin, J., Design of Real-Time Computer Systems, Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1967.

(26) McGregor, P., "Load Sharing in a Computer Network", Ph.D. Thesis, Dept. of Electrical Engr., Polytechnic Institute of New York, June, 1974.

(27) McGregor, P. and Boorstyn, R.R., "Load Sharing in a Computer Network", International Conference on Communications, San Francisco, June 16-18, p. 41-14 to 41-19, 1975.

(28) Metcalfe, R.M., "Packet Communication", Project MAC Report TR-114, M.I.T., Dec. 1973.

(29) Moore, C., "Network Models for Large-Scale Time-Shared Systems", Ph.D. Thesis Dept. of Industrial Engineering, U. of Michigan, April 1971.

(30) Roberts, L.G. and Wessler, B.D., "Computer Network Development to Achieve Resource Sharing", Spring Joint Computer Conference, AFIPS Conference Proceedings, Vol. 36, 1970, pp. 543-549.

(31) Roome, W.D. and Torng, H.C., "Modeling and Design of Computer Networks with Distributed Computer Facilities", Proceedings of the 1974 Symposium, Computer Networks: Trends and Applications, IEEE Computer Society, 1974.

(32) Rubin, I., "Message Path Delays in Packet-Switching Communication Networks", IEEE Transactions on Communications, Vol. 23, No. 2, Feb. 1975, pp. 186-192.

(33) Saaty, T.L., Elements of Queueing Theory, New York: McGraw-Hill, 1961.

(34) Strom, C.A., Jr. and Walker, R.K., "Distributed Computer-Communication Networks", Proceedings of the Symposium on Computer-Communication Networks and Teletraffic, Brooklyn, N.Y.: Polytechnic Press, 1972.

(35) Wilkov, R.S., "Analysis and Design of Reliable Computer Networks" IEEE Transactions on Communications, Vol. 20, No. 3 Part II, June 1972, p. 660.

Distribution List

Defense Documentation Center Cameron Station Alexandria, Virginia 22314	12 copies
Assistant Chief for Technology Office of Naval Research, Code 200 Arlington, Virginia 22217	1 copy
Office of Naval Research Information Systems Program Code 437 Arlington, Virginia 22217	2 copies
Office of Naval Research Code 1021P Arlington, Virginia 22217	6 copies
Office of Naval Research Branch Office, Boston 495 Summer Street Boston, Massachusetts 02210	1 copy
Office of Naval Research Branch Office, Chicago 536 South Clark Street Chicago, Illinois 60605	1 copy
Office of Naval Research Branch Office, Pasadena 1030 East Green Street Pasadena, California 91106	1 copy
New York Area Office (ONR) 715 Broadway - 5th Floor New York, New York 10003	1 copy
Naval Research Laboratory Technical Information Division, Code 2627 Washington, D.C. 20375	6 copies

Dr. A. L. Slafkosky — 1 copy
Scientific Advisor
Commandant of the Marine Corps (Code RD-1)
Washington, D.C. 20380

Office of Naval Research — 1 copy
Code 455
Arlington, Virginia 22217

Office of Naval Research — 1 copy
Code 458
Arlington, Virginia 22217

Naval Electronics Laboratory Center — 1 copy
Advanced Software Technology Division
Code 5200
San Diego, California 92152

Mr. E. H. Gleissner — 1 copy
Naval Ship Research & Development Center
Computation and Mathematics Department
Bethesda, Maryland 20084

Captain Grace M. Hopper — 1 copy
NAICOM/MIS Planning Branch (OP-916D)
Office of Chief of Naval Operations
Washington, D.C. 20350

Mr. Kin B. Thompson — 1 copy
Technical Director
Information Systems Division (OP-91T)
Office of Chief of Naval Operations
Washington, D.C. 20350

Advanced Research Projects Agency — 1 copy
Information Processing Techniques
1400 Wilson Boulevard
Arlington, Virginia 22209