INFORMATION TO USERS

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

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

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

2. When an image on the film is obliterated with a large round black mark, it is an indication that the photographer suspected that the copy may have moved during exposure and thus cause a blurred image. You will find a good image of the page in the adjacent frame.

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

4. The majority of users indicate that the textual content is of greatest value, however, a somewhat higher quality reproduction could be made from "photographs" if essential to the understanding of the dissertation. Silver prints of "photographs" may be ordered at additional charge by writing the Order Department, giving the catalog number, title, author and specific pages you wish reproduced.

5. PLEASE NOTE: Some pages may have indistinct print. Filmed as received.

University Microfilms International
300 North Zeeb Road
Ann Arbor, Michigan 48106 USA
St. John's Road, Tyler's Green
High Wycombe, Bucks, England HP10 8HR
BAIN, WILLIAM LAMAR, JR.,
HARDWARE SCHEDULING STRATEGIES FOR SYSTEMS
WITH MANY PROCESSORS.

RICE UNIVERSITY, PH.D., 1978
RICE UNIVERSITY

HARDWARE SCHEDULING STRATEGIES
FOR SYSTEMS WITH MANY PROCESSORS

by

WILLIAM LAMAR BAIN, JR.

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

DOCTOR OF PHILOSOPHY

THESIS DIRECTOR'S SIGNATURE:

[Signature]

HOUSTON, TEXAS

JANUARY, 1978
ACKNOWLEDGEMENTS

I wish to thank Dr. J. Robert Jump, my thesis director, for his guidance in this research. Dr. Edward A. Feustel and Dr. Paul Pfeiffer are to be thanked for serving on my thesis committee.

My friends Sudhir Ahuja, Tim Gonsalves, Anand Jagannathan, Rajan Kapur, and Rick White have my gratitude for making my graduate life enjoyable.

Debbi Howe deserves special thanks for her careful and competent typing of the manuscript and for her patience and concern. And I thank Mrs. Goodwin, who takes care of us all.

Finally, I express my sincere appreciation to Candy Roberts whose unselfish assistance, encouragement, and caring I will always remember.

This research was supported by National Science Foundation Grant DCR74-14283.
# TABLE OF CONTENTS

## CHAPTER I  INTRODUCTION AND BACKGROUND

1-1  Introduction  
1-2  Background  

## CHAPTER II  DESCRIPTION OF THE SYSTEM AND ANALYSIS TECHNIQUE

2-1  Description of the System  
2-2  Assumptions for the Analysis  
2-3  Performance Measures  
2-4  The Single Processor System  
2-5  Systems with N Processors  

## CHAPTER III  INTERCONNECTION NETWORKS WITHOUT BUFFERS

3-1  Introduction  
3-2  The Ordered Round Robin System  
3-3  The Unordered Round Robin System  
3-4  The Left Priority System  

## CHAPTER IV  INTERCONNECTION NETWORKS WITH BUFFERS

4-1  Introduction  
4-2  The Loop System  
4-3  The Hybrid System  

## CHAPTER V  APPLICATIONS, EXTENSIONS, AND CONCLUSIONS

5-1  Applications  
5-2  Extensions  
5-3  Suggestions for Further Research  
5-4  Conclusions
APPENDIX

REFERENCES
CHAPTER I
INTRODUCTION AND BACKGROUND

1-1 Introduction

This thesis studies the use of large numbers of processors in a multiprocessor system. The processors are organized into a special functional unit, called a "backend machine," within a general computer system. The system's central processor (CPU), which handles user and systems programs as well as input/output, is connected to the backend machine by two data busses. (See figure 1-1.) The CPU delivers a stream of independent tasks in the form of "packets" to the backend machine which executes the tasks concurrently and returns the results back to the CPU. Modular hardware techniques for the effective distribution of these packets to the processors are presented, and their performance is modeled and compared. It is shown that the addition of hardware buffering for the packets significantly improves the performance of the system.

The motivation for this multiprocessor organization and its relationship to other architectures are studied in this chapter. The inadequacy of existing analytical techniques for measuring the performance of these systems is also demonstrated. In chapter two, the structure and operation of the backend machine is presented in detail. Simplifying assumptions required for its probabilistic modeling are given, and the performance measures are defined. The modeling technique is illustrated for a trivial system with one processor. Preliminary observations applicable to all systems with N processors are also presented. The hardware scheduling techniques
FIGURE 1-1: System Block Diagram
are divided into two classes, those without and those with buffering for the packets, and these two classes are studied separately in chapters three and four, respectively. In the systems discussed in chapter three, the processors are connected directly to the two busses. Two "round-robin" methods for bus arbitration are analyzed and compared to the commonly used "left-priority" technique. These systems serve as a benchmark for evaluating the effect of buffering on performance. The first system with buffering discussed in chapter four, called the "loop system," has one buffer for every processor, the maximum number of buffers used in these systems. The second system, called the "hybrid system," has an arbitrary (but smaller) number of buffers and combines aspects of both the round-robin and loop systems. It represents a design methodology by which desired performance tradeoffs can be obtained with a suitable amount of buffering. In chapter five, the incorporation of multiple packet streams into these architectures is examined. Also, a viable application of this organization as a dedicated data-base machine is given, and possible directions for further research are suggested.
1-2 Background

With the advent of large scale integration (LSI) technology, it has become feasible to consider multiprocessing systems with large numbers of processors and memories. Unfortunately, many of the concepts used in the design of conventional multiprocessing systems with relatively few (i.e. less than sixteen) powerful processors, such as the systems discussed by Enslow [ENSL 74] do not extend readily to this new domain. In this section, we discuss some of the restrictions imposed by the presence of a large number of processors that motivate the processor interconnection and scheduling schemes presented and analyzed in this thesis. We show how the approach adopted here differs from alternative designs and indicate the importance and nature of its performance analysis.

One of the more important and difficult problems in the design of multiprocessing systems is the decomposition of algorithms into a form suitable for concurrent processing [SIEW 76]. Only for a few classes of problems, such as sorting [PREP 77] and the solution of linear recurrence relations [KUCK 75], has the potential for speedup through the exploitation of parallelism been explored. Mechanisms for the explicit specification of available parallelism in programs are not well developed or in widespread use, and the automatic detection of parallelism has proven to be very difficult [BAER 73]. Systems with few processors have successfully handled this problem by dedicating processors to certain functions such as I/O processing (e.g. Xerox Sigma 9) and/or through multiprogramming and the a priori segmentation of the operating system and user
programs [ENSL 74]. The magnitude of this problem increases dramatically for systems with many processors. One solution is to dedicate the system to executing programs of known and plentiful concurrency. For instance, array and associative processors (eg. ILLIAC IV [BARN 68] and STARAN [ENSL 74], respectively) execute one algorithm concurrently on several data. These single instruction stream-multiple data stream (SIMD) machine (using Flynn's classification scheme [FLYN 72]) often do not have full processors operating on each datum and hence are highly restricted in the algorithms which they execute effectively. Ahuja [AHUJ 77] has shown how a vector processor can be constructed out of a large number of microprocessors in such a manner that it is suitable for other applications, such as algorithms based on the perfect shuffle [STON 71]. In his system, each processor holds a fixed set of programs in its local memory; only control information and data are passed to the processors. Similarly, Cooper uses a linear arrangement of microprocessors to behave like a generalized pipeline [COOP 77]. By dedicating a system with many processors to a fixed class of problems, a large pool of concurrent tasks is potentially made available. Such a machine is useful as a special purpose functional unit in a general computing system or computer communications network. We adopt this approach for the systems discussed in this thesis.

A second problem relating to the design of a multiprocessor system is the interconnection of the processors and memories. Conventional general-purpose multiprocessors usually interconnect
these components through either a time shared bus (or set of busses), a multiport memory scheme, a crossbar switch, or some combination of these techniques [WEIS 77]. See figure 1-2. The time shared bus has the worst performance, although processors and memories are easily added or removed. With a large number of processors, its performance deteriorates badly in general, as will be shown in chapter three. The multiport memory scheme lacks expandability since there are a fixed number of memory ports; the cabling costs are also very high. An N x N crossbar switch has a cost of $O(N^2)$ which is prohibitive for large N. Lawrie [LAWR 75] and Lang [LANG 76] have reduced this cost to $O(N \log N)$ at the expense of the generality of allowed interconnections and additional time required to perform the transfers. Normally systems with few processors interleave the memories in order to balance the number of requests presented to each memory. Baskett and Smith [BASK 76] and Rau [RAU 76] have shown that the bandwidth of an interleaved memory system (under certain ideal assumption) is highly impaired in a system with as few as eight processors and memories due to contention for the memories. Smith [SMIT 77] and Hoogendoorn [HOOG 77] demonstrate that the contention is greatly reduced when the processors use local memory predominately; and this is the trend for some current multiprocessors, such as the CM* at Carnegie-Mellon University [FULL 77]. The use of local memory can also simplify the interconnection network by allowing,

*Sullivan and Bashkow [SULL 77] dispute the necessity to use local memory and propose a system with $10^5$ to $10^6$ processors and memories using interleaved memory.
FIGURE 1-2: Processor-Memory Interconnection Schemes
P = processor, M = memory module
for example, a hierarchical arrangement, and its importance is magnified with a large number of processors in the system. The restriction to a special purpose architecture facilitates its implementation since the memory reference patterns are well known. In this thesis, we assume that all processors execute instructions exclusively from a fixed set of algorithms held in their local memories. It may be necessary to access global data, however.

Processor scheduling is a third important problem. It is well known that the optimal non-preemptive scheduling of \( T \) independent tasks on more than two processors so as to minimize the total schedule length is an NP-complete problem [COFF 76]. The use of preemption reduces this problem to \( O(T \cdot \log_2 T) \) complexity [COFF 76], and the "largest processing time first" (LPT) schedule yields near optimal results for large \( T \) and restricted maximum and minimum task execution times [IBAR 77]. The implementation of these algorithms may be impractical however, especially for a large number of processors. And the task execution times may not be known in advance, so that the processors may become available at unpredictable times. The processors can either be scheduled by a centralized control unit (eg. the CDC 6600 [ENSL 74]) or by the processors themselves (eg. the Burroughs Multi-Interpreter [DAVIS 72]). Gonzalez and Ramamoorthy [GONS 72] compare the performance of these two approaches through simulation. In either case, if tasks are scheduled sequentially, as is commonly done, this leads to a serious deterioration in systems with many processors and random task execution times, as shown in chapter three. Arnold [ARNO 76], Levy [LEVY 73] suggest that the processors
schedule themselves hierarchically. This method is only suitable for certain classes of algorithms and may involve a large software overhead. In this thesis, we present efficient hardware schemes for concurrently scheduling large numbers of processors. This is accomplished by dispensing the control information required to initiate tasks in the form of packets which are routed to the processors through a system of buffers. We assume that the packets issue in a single stream from a central processor to which this system is connected; the processors return packets indicating the completion of their tasks back to the central processor in another single stream. The use of a single packet stream allows a simple interconnection to the central processor.* Ahuja [AHUJ 77] proposed this architecture and demonstrated its effectiveness in vector processing where all tasks have nearly identical execution times. We investigate its performance under the more general assumption that the tasks' execution times may vary randomly according to a known (but arbitrary) probability distribution function, as might be the case in other applications. We find that by appropriately buffering the packets, very high performance can be obtained. The software requirements for the formulation and coordination of the packets by the central processor are not studied here. It is assumed that the nature of the application for which the system is intended (e.g. its use as a dedicated database management system, as studied in chapter five) facilitates these activities.

*In chapter five we suggest how this architecture might be generalized to accommodate multiple streams.
The performance analysis of computing systems is a difficult and yet important aspect of their design, for it allows the quantitative comparison of competitive systems. Unfortunately, there are no standard performance measures or analysis techniques. Ramamoorthy and Krishnarao [RAMA 76] suggest several criteria, including system throughput, response time, resource utilization, and cost. Chow and Kohler [CHOW 76] use throughput and turnaround time as the principal performance criteria of their queuing models for multiprocessing systems. Each of the systems discussed in this thesis is either modeled probabilistically (using elementary Markov chains [PARZ 60]) or simulated in PL/I to determine the average throughput and turnaround time in the system of the tasks. It is desirable that the throughput of an N processor system approach N-fold that of a single processor system, although this goal is never fully reached. It is also desirable that the systems be modular so that processors and memories can be easily added or removed from the system without changing its basic structure. Such a system offers incremental improvement in performance with the simple addition of processors and graceful degradation in the event of processor failure.

It should be noted that the systems discussed here are different from the computer communication networks such as Farber's distributed computing system [FARB 72], and the loop networks of Pierce, Newhall and Reames and Liu [REAM 76] in that no inter-processor communication is allowed. The processors communicate solely with the external central processor, and the interconnection networks
serve only as a mechanism for the "load-sharing" of tasks among them. Hence, the analyses relative to the latter systems are inapplicable here.
CHAPTER II
DESCRIPTION OF THE SYSTEM AND ANALYSIS TECHNIQUE

2-1 Description of the System

A block diagram of the single stream special purpose system, is shown in figure 2-1. Tasks are formulated by the central processor in the form of input packets and placed sequentially in the input register. The input packets are routed to available processors via an interconnection network consisting of data paths and perhaps additional registers, and a hardware control mechanism. The processors execute the tasks specified by the input packets and generate output packets upon completion. The output packets are placed sequentionally in the output register in order that the results may be returned to the central processor. The tasks thus "stream" through the system from the input to the output registers. The registers can be incorporated into the central processor or included in the system; they represent the storage elements where the packets sequentially reside until withdrawn by the system or the central processor and are useful in defining performance measures for the system. A register can hold only one packet at a time. The data paths interconnecting registers and processors may be of any width; the choice affects the cost of the system and its resulting performance.

As indicated earlier, the system executes fully independent tasks; there is no interprocessor communication during execution. Hence, the central processor is required to issue only those input packets to the system for which there are no precedence constraints
FIGURE 2-1: Detailed System Block Diagram
among their respective tasks. These packets contain all of the necessary information required to initiate execution of the tasks. This information includes:

1. task identification,
2. program identification,
3. input data.

The task identification is passed along to the output packets by the system's processors; it is required by the central processor since, in general, the order of input packets entering the system is not preserved in the stream of corresponding output packets. The program identification allows the processors to select the appropriate program for execution from a set of available programs stored in the processors' local memories. This reduces the substantial communications overhead that would be required to pass the code for each task to the processors in the system.

The input data are those data required by the individual tasks or pointers to the actual data contained in a central memory if the amount of data is very large. The output packets generated by the processors contain the task identification and output data. As with the input packets, the output data may actually be pointers to data deposited by the processors in the central memory.

This system is a "load distributed" system as opposed to a "functionally distributed" system. That is, any task may be executed by any processor; the processors are not specialized in the tasks that they may execute. The routing of input packets to available processors and the corresponding routing of output packets
to the output register is accomplished by the interconnection network and a hardware control mechanism. It is desirable that these components be divided up among the processors in the machine (and perhaps included on the same LSI chip). In this way, systems of increasing performance can be constructed by adding processor/network/control modules. Such a configuration is "fail-soft", as well, in the sense that failure of a module does not imply failure of the entire system if the failing module can be isolated and removed from the system. The performance is simply degraded to that of a system with one less module.
2-2 Assumptions for the Analysis

Several assumptions will be imposed on the operation of the system to facilitate the analysis of its performance. First, it is assumed that packets propagate among the registers and processors synchronously with a central system clock. The period of this clock is the total time required for one bit of data or control information to be transferred from a register or processor to a neighboring register or processor. This is determined by the technology of the devices employed (i.e., the gate delays), and the actual transmission time. Packet length is denoted by positive integer $L$. These lengths might actually tend to vary from task to task and between input and output; the registers in the interconnection network would have to accommodate "worst-case" packet lengths, however. The data paths interconnecting these registers and connecting them to the processors are $W$ bits wide by definition. Whereas the parameter $L$ is determined by the nature of the tasks being executed, the parameter $W$ is dependent on the tradeoff between the cost of the data paths and the performance obtained. The total transfer time, $r$, for a packet then equals $\left\lfloor \frac{L}{W} \right\rfloor$ clock periods. In fact, such transfers could and should be asynchronous to avoid problems such as clock skew. The above assumptions provide at least a framework for the study of actual systems.

We assume that there are no bottlenecks imposed by the central processor. More specifically, a new input packet is always
transferred to the input register concurrently with the removal of the previous one. Further, the central processor removes every packet from the output register immediately after it arrives there. This assumption causes the analysis to yield an upper bound on the performance of the system by isolating it from the characteristics of the central processor. It greatly simplifies this analysis as well.

A final assumption is that the execution times for the tasks vary randomly according to a specified but arbitrary discrete probability mass function, \( P'(t) \), and corresponding distribution function \( P(t) \), where \( t \) varies from one to \( h \) clock periods for some positive integer \( h \). The average task execution time, \( p^* \), is defined in the usual manner:

\[
p^* = \sum_{t=1}^{h} t \cdot P'(t)
\]

Since the tasks are independent, then the input packets entering the system have execution times that are independent and identically distributed according to \( P'(t) \). The communication delays between the processors and a central memory can be incorporated into this distribution if it is possible to model them probabilistically. Baskett and Smith [BASK 76] show how this can be accomplished for a system of \( N \) processors fully interconnected to \( M \) memory modules under certain approximations.

\[\text{The registers are functionally } L \text{ r-bit shift registers connected in parallel.}\]
2-3 Performance Measures

In order to assess the steady-state performance of the system, i.e., the average performance after the system has been in operation for very many (approaching infinity) clock periods, the following measures are used:

(1) the average throughput of the system, \( TP \), measured in the number of tasks completed per clock period, and

(2) the average turnaround time, \( TT \), for the tasks executed by the system, measured in clock periods.

Assume that \( N_T \) output packets reach the output register after the system has been in operation for \( T \) clock periods. Then we have:

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

Note that \( 0 \leq TP \leq 1/r \).

The turnaround time for a given task is defined as the number of clock periods required for a task to reach the output register (as an output packet) after its arrival in the input register (as an input packet). Assume that the \( i^{th} \) task that reaches the output register has a turnaround time of \( TT(i) \) clock periods. Then we have:

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

These performance measures are calculated as a function of the number of processors in the system, \( N \), the parameters \( L \) and \( W \), and the probability mass function of execution times, \( P'(t) \).
These functions depend on the interconnection network and control mechanism used to connect the processors to the input and output registers.
The Single Processor System

The approach to the analysis of the various configuration of this system will be to describe its behavior in time in terms of a relevant set of states through which it passes (synchronously with the system clock). The probability that the system is in a given state is determined for every state as a function of the parameters \( N, L, W, \) and \( P \). The performance measures are then formulated from the state probabilities. The mathematics of discrete time Markov chains is employed to determine these state probabilities. Often a Markov chain is developed for each processor; in this case the states represent whether the processor is busy or idle. The single processor system will be used to illustrate this technique.

Consider a system with one processor, as shown in figure 2-2; the processor is connected directly to the input and output registers. Assume that the data paths are \( L \) bits wide, i.e., \( L = W \). Hence, packets are transmitted in parallel within one clock period \((r = 1)\). Suppose the task execution times are distributed geometrically (the discrete time memoryless distribution) with parameter \( p \). That is:

\[
p'(t) = p \cdot (1-p)^{t-1}, \quad t = 1, 2, 3, \ldots
\]

The average task execution time, \( p^* \), is just \( \frac{1}{p} \) in this case. Two states for the processor can be defined: "idle," in which the processor is concurrently receiving an input packet from the input register and delivering an output packet to the output register, and "busy," in which the processor is executing a task. (It is
FIGURE 2-2: Block Diagram of the Single Processor System
assumed that processors can concurrently input and output packets.)
A graphical representation of these states in the form of a
Markov chain, along with the state transition probabilities, is
shown in figure 2-3. The processor moves from the idle state to
the busy state with probability \( t_0 \), which equals unity since the
input register is always full. It begins execution and remains
busy with constant probability \( 1-p \), regardless of how long it has
been in execution (the memoryless property); it completes execution
and returns to the idle state with probability \( p \). It remains
in the idle state for exactly one clock period during which time
the output packet for the completed task is placed in the output
register and a new input packet is withdrawn from the input register.
See figure 2-4. The probability that the processor is in the idle
state, \( \lambda \), is determined by solving the trivial linear equations:

\[
\lambda = p \beta + (1-t_0) \lambda, \quad \text{and}
\]

\[
1 = \lambda + \beta,
\]

where \( \beta \) is the probability that the processor is in the busy state.
Note that the second (constraint) equation assures that the total
probability that the processor is in some state equals unity. This
leads to the following result:

\[
\lambda = \frac{1}{1+t_0 p^*} = \frac{1}{1+p^*}
\]

since \( p^* = \frac{1}{p} \), and \( t_0 = 1 \). This Markov chain meets the necessary
requirements for ergodicity [KLEI 75], and thus the above state
probabilities do indeed approach stable values. Since an output
FIGURE 2-3: Markov Chain for the Single Processor System with Task Execution Times Geometrically Distributed
System Clock:

```
...----|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--||
packet is placed in the output register at every clock period in which the processor is in the idle state (except the very first), then it follows from the Law of Large Numbers that

\[ TP = \frac{1}{1+p^*} \]

The average turnaround time for this system is easily derived. It consists of three components:

- \( T_{T_i} \), the average time that a task spends in the system as an input packet,
- \( T_{T_e} \), the average time that a task spends in execution, and
- \( T_{T_o} \), the average time that a task spends in the system as an output packet.

The average turnaround time is just the sum of these components:

\[ TT = T_{T_i} + T_{T_e} + T_{T_o} \]

since the sum of averages equals the average of sums. Now, every task resides in the input register (as an input packet) for exactly the length of time that the previous task requires to execute since (1) it arrives in this register at the clock period in which the previous task is withdrawn by the processor, and (2) it remains there until execution of the previous task is completed. Hence,

\[ T_{T_i} = p^* + 1 \]

where the extra clock period accounts for the inputting of the task to the processor. Also, we have trivially:

\[ T_{T_e} = p^* \quad \text{and} \quad T_{T_o} = 1 \]
where the latter result follows from the fact that the output register is always available to the processor. Hence, the total turnaround time is just
\[ \text{TT} = 2(p^* + 1) = \frac{2}{TP} . \] (2-1)

The throughput and turnaround time can be calculated in the more general case in which the task execution times are specified by an arbitrary probability mass function \( P'(t) \). A state diagram containing \( h + 1 \) states as shown in figure 2-5, is constructed for this purpose. The processor is either idle or in one of the \( h \) busy states; it is busy state \( i \) if and only if the task is in the \( i^{th} \) clock period of execution. The processor moves from the idle state to busy state 1 upon receipt of a new task; this occurs with probability \( t_0 = 1 \), as before. After one clock period of execution, it propagates to busy state 2 if the task is of length 2 or greater. In general, the processor will move from busy state \( i \) to busy state \( i + 1 \) after \( i \) clock periods of execution if the task is at least of length \( i + 1 \) in duration, given that the task is at least of length \( i \). The transition probability \( t_i \) is thus expressed as a conditional probability:

\[ t_i = p \{ \text{the task is of length } i + 1 \text{ or greater } | \text{the task is at least of length } i \}. \]

In terms of the distribution function \( P(t) \), we have:
\[ t_i = \frac{1-p(i)}{1-p(i-1)} , \quad i = 1, \ldots, h . \] (2-2)
The processor returns to the idle state from busy state \( i \) with probability \( 1 - t_i \) since the total probability that it leaves busy state \( i \) must sum to unity. Note that \( t_n = 0 \), and so the processor always returns to the idle state after \( h \) clock periods of execution, as expected.

As before, the throughput and turnaround time can be determined from the idle state's state probability \( \iota \). This requires a solution of \( h + l \) linear equations in the \( h + l \) unknowns \( \iota, \beta_1, \beta_2, \ldots, \beta_h \) where \( \beta_i \) is the state probability for busy state \( i \):

\[
\begin{align*}
\beta_1 &= t_0 \iota, \\
\beta_2 &= t_1 \beta_1, \\
\beta_3 &= t_2 \beta_2, \\
\vdots & \quad \vdots \\
\beta_h &= t_h \beta_{h-1}, \\
\iota &= \iota + \beta_1 + \beta_2 + \ldots + \beta_h.
\end{align*}
\]

This system has a particularly trivial solution:

\[
\iota = \frac{1}{1 + \sum_{i=0}^{h-1} \sum_{j=0}^{\pi} t_j} \tag{2-3}
\]

Plugging in the transition probabilities using equation (2-3), we have:

\[
\begin{align*}
\iota &= \frac{1}{1 + t_0 \left( 1 + \sum_{i=1}^{h-1} \left[ 1 - P(i) \right] \right)} \\
&= \frac{1}{1 + t_0 \left( 1 + \sum_{i=1}^{h-1} \left[ i \cdot P'(i+1) \right] \right)}
\end{align*}
\]
\[ l = \frac{1}{1 + t_0 \left( \sum_{i=1}^{h-1} P'(i) + \sum_{i=1}^{h-1} i \cdot P'(i+1) \right)} \]

\[ = \frac{1}{1 + t_0 \cdot p^*} \quad (2-4) \]

\( l \) is of course identical to the previous value when the task execution times are geometrically distributed; hence the throughput and turnaround time are the same as before, regardless of the distribution function. Finally, we have:

\[ TP = \frac{1}{l + p^*} = \frac{2}{11} \]

for the single processor system for task execution times distributed arbitrarily with average \( p^* \).

The method of analysis employed for the single processor system will be used for all of the following multiple processor systems with various interconnection networks and control mechanisms.

The above results can be generalized to allow for arbitrary packet transfer times. As indicated above, a single processor system with \( r = 1 \) can execute tasks at the average rate of one task per \( p^* + 1 \) clock periods if tasks are continuously made available to it, as they are in this case. In general, the processor can at most execute one task per \( p^* + r \) clock periods for arbitrary packet transfer time \( r \). Thus, we have:
\[ TP = \frac{1}{p^* + r}, \quad \text{and} \]  
\[ TT = 2(p^* + r) = \frac{2}{TP} \]  
(2-5) (2-6)

for the single processor system with arbitrary \( r \). The turnaround time is based on the same reasoning as before; we have:

\[ TT_i = p^* + r, \]
\[ TT_e = p^*, \quad \text{and} \]
\[ TT_o = r, \]

where \( r \) clock periods are required to input and output the packets (instead of one clock period).
2-5 Systems with N-Processors

A system with N processors should ideally exhibit N-fold the throughput of a single processor system if all processors have continuous access to the input and output registers (i.e., if they never waste clock periods waiting for this access). This is equivalent to a throughput of \( \frac{N}{p^* + r} \) tasks per clock period. Clearly, if two or more processors become available at the same time, only one processor can receive access to the input and output registers. That is, the maximal throughput that a single-stream can deliver is \( \frac{1}{r} \) tasks per clock period. If

\[
\frac{N}{p^* + r} < \frac{1}{r}
\]

then the system is limited by the number of processors in the system; if

\[
\frac{1}{r} < \frac{N}{p^* + r}
\]

then the system is limited by having a single stream of communication to the control unit. Hence, we have

\[
TP \leq \min \left( \frac{1}{r}, \frac{N}{p^* + r} \right)
\]

A graph of the ideal throughput versus the number of processor, \( N \), in a single stream system with \( p^* = 55.5 \) and \( r = 1 \) is shown in figure 2-6.

The above relation is an inequality because the throughput of actual systems is often constrained by the following considerations. First, all processors may be busy at clock periods when the input
and output registers are available. It can be shown that for a system with \( N_h = \lceil \frac{h+r}{r} \rceil \) processors (where \( h \) is the maximal allowed task execution time), there exists a control mechanism for which this will never occur; i.e., there will always be a free processor whenever the input/output registers are available.* Furthermore, the control mechanism always locates a free processor without delay (see below) after a transfer is completed. Hence, this system achieves the maximal throughput of \( \frac{1}{r} \), with a very large number of processors. Ideally, a system could achieve this throughput with as few as \( N_\ell = \lceil \frac{p^*+r}{r} \rceil \) processors if it exhibits \( N \)-fold the throughput of a single processor system (except as limited by the single stream). Second, a processor may be ready to accept new tasks and the registers may be idle, but the control mechanism fails to locate the processor immediately. The control mechanism is designed so as to minimize the average time, \( t_f \), required to locate an available processor once a transfer is completed. Its impact on throughput is expressed as:

\[
TP = \frac{1}{t_f + r}
\]  

(2-7)

where \( t_f = 0, 1, 2, \ldots \) clock periods. That is, a task can be issued once every \( t_f + r \) clock periods on the average.

*See Ahuja [AHUJ 77]. If we assume that all tasks require the longest possible execution time, \( h \) clock periods, then his system achieves \( TP = \frac{1}{r} \) with \( N_h \) processors. His performance measure is the time, \( T_z \), required for the completion of \( z \) tasks. Its relationship to \( TP \) is:

\[
TP = \lim_{{z \to \infty}} \left( \frac{z}{T_z} \right).
\]
The average value for that part of the tasks turnaround time, $TT_W$, during which the tasks wait to be output from the processors after execution is completed can be calculated as follows. This calculation is based on the aforementioned assumption that tasks are continually available from the central processor and the assumption that, as a consequence, new tasks are supplied to processors whenever output packets are withdrawn. The latter assumption holds for all systems described in this thesis, although it may not hold in general. These assumptions imply that processors, once initially loaded with tasks, are never idle without holding completed tasks waiting for output. Hence, $TT_W$ is simply the total idle time accumulated by all of the processors, divided by the total number of tasks completed, $N_T$. Now, $N$ processors are in execution or (concurrent) input/output for $N_T(p^* + r)$ of the $N\cdot T$ possible clock periods available to them during the $T$ clock periods of real time that the system is active. Thus, we have:

$$TT_W = \lim_{T \to \infty} \frac{N\cdot T - N_T(p^* + r)}{N_T}$$

$$= \frac{N}{TP} - (p^* + r) \quad \text{clock periods.} \quad (2-8)$$

Note that $TT_W$ is always non-negative and equals zero when $\frac{N}{TP} = p^*r$, i.e. when:

$$TP + \frac{N}{p^* + r}$$

That is, the processors are never idle (once initially loaded) when the system exhibits $N$-fold the throughput of a single processor system.
The above formula for $TT_W$ is used in the calculation of turnaround time for the systems discussed herein.
CHAPTER III
INTERCONNECTION NETWORKS WITHOUT BUFFERS

3-1 Introduction

Perhaps the simplest interconnection network for a multiprocessor system with \( N \) processors is shown in figure 3-1. It consists simply of two data busses (\( W \) bits wide), one connecting the processors to the input register and one connecting them to the output register. The control mechanism for arbitrating the busses can be represented in part by \( N \) flip-flops, one for each processor, with exactly one flip-flop at logical "one" during any clock period. At that time, the processor associated with the flip-flop has access to both the input and output registers. If it has completed execution of the current task, it concurrently outputs an output packet and obtains a new input packet, this action requiring \( r \) clock periods. The control "bit" remains at the processor's flip-flop throughout the duration of the transfer, and propagates to another flip-flop during the last clock period of the transfer. We will assume that one control mechanism arbitrates both busses since the processors require access to both busses when they complete their respective tasks.

The remaining part of the control mechanism determines the order in which the processors receive access to the busses. Three schemes will be considered here, and their performance compared. The first two are "round-robin" systems in which the processors are scheduled in a circular manner. They differ in the condition required for the control bit to move from a processor
FIGURE 3-1: The Round-Robin System
to its successor. In the first case, the bit moves from one processor to the next only after the processor both completes execution of its current task and receives a new task from the input register. In the second case, the bit moves continuously from one processor to the next until a processor requiring access to the busses (i.e., a processor that has completed its current task) is located.

The third scheme is the so-called "left priority first" mechanism used in many actual systems (e.g., PDP UNIBUS). The processors are given access to the busses according to a static set of priorities, usually determined by their relative location on the busses. These control mechanisms can all be distributed among the processors so that the resulting system is fully modular, as will be seen below.

The above interconnection networks are characterized by the lack of intermediate registers, or buffers, to hold input and output packets within the network. They provide a benchmark for the study (in chapter 4) of the effect that such buffers have on the performance of the system.
3-2 Ordered Round Robin System

This system was used very effectively by Ahuja in a special purpose system for vector processing [AHUJ 77]. In this system the control bit only moves from a processor to its successor after the processor has completed its current task and performed the input/output required to start a new one. An example of the behavior of this system is shown in figure 3-2 with \( N = 3 \) and \( r = 2 \). The control bit initially resides at processor 1. During clock periods 1 and 2, a task with execution time equal to 6 is transferred into the processor. At clock period 2 the bit moves on to processor 2. A task with execution time equal to 3 is transferred to this processor during clock periods 3 and 4. The bit moves to processor 3 at clock period 4, and a task of length 8 is transferred to this processor at clock periods 5 and 6. The control bit returns to processor 1 during clock period 6 and waits there until processor 1's task is completed. At clock periods 9 and 10 the processor outputs its results for task 1 and concurrently inputs task 4. The bit moves to processor 2 at time 10 and finds that processor 2 has already completed task 2. Thus, processor 2 performs its input and output of packets at clock periods 11 and 12, and so on.

This system has the distinct advantage of maintaining the original order of the input stream of tasks in the resulting output stream, hence the name "ordered" round-robin (ORR) system. It behaves much like a pipeline in this respect. In fact, Ahuja was able to prove that this system exhibits the same (or less) time to complete \( Z \) tasks as an equivalent \( N \)-stage pipeline; under the
FIGURE 3-2: Illustration of the ORR System's Behavior
condition that all task execution times are fixed at some h clock periods. It has the added advantage that the processor can be programmed to execute an arbitrary algorithm. Also, the system is highly modular; incremental improvement or degradation in performance is obtained by adding or removing modules, without changes in the algorithms or the structure of the interconnection network.

The method of analysis for the ORR system will be to model with a Markov chain the sequence of delays encountered by the control bit as it circulates around the control loop. The average delay, \( t_f \), that the bit suffers as it services each processor is calculated from the state probabilities of this chain. The performance measures, throughput and turnaround time, follow from this parameter.

Let \( d_1, d_2, \ldots, d_n, \ldots \) represent a sequence of such delays, where \( d_n \) is the \( n^{th} \) delay encountered since the system initiated operation, for positive integers \( n = 1, 2, 3, \ldots \); this delay occurs at processor \( 1 + R \lfloor (n-1)/N \rfloor \) where \( R[p/q] \) denotes the remainder of the division of \( p \) by \( q \). These delays are non-zero when the control bit arrives at a processor that has not completed execution of its current task; the \( r \) clock periods required for the input/output of packets are not included in the delays. Two constraints on the delays which serve as the basis for the construction of the Markov chain can be stated. First no delays can be more than \( M \equiv h \geq (N-1)r \) clock periods long where \( a \geq b \equiv \text{maximum of } h - (N-1)r \) and zero (negative delays are not allowed); this follows since
(a) tasks have execution times not larger than \( h \) clock periods
and (b) at least \((N-1) \cdot r\) clock periods are required for the control
bit to return to a processor after the current task was input.
Hence, we have: \( 0 \leq d_n \leq M \) for \( n = 1, 2, 3, \ldots \). Second, once the
bit has accumulated a total delay of \( M \) clock periods on any circuit
around the loop, then none of the remaining processors can delay
the bit; that is, every contiguous sequence of \( N \) delays has a
total of at most \( M \):

\[ \sum_{j=0}^{N-1} d_{n+j} \leq M \quad \text{for} \quad n = 1, 2, 3, \ldots \]

**Theorem 3-1:**

**Proof:** This can be proved by induction on \( n \), the number of
processors serviced since the initiation of the system. (Basis
step, \( n = 1 \)). Clearly, the first \( N \) delays encountered by the
control bit: \( d_1, d_2, \ldots, d_N \) are precisely zero since all processors
are initially idle. Hence we have \( \sum_{j=0}^{N-1} d_{1+j} = 0 \leq M \).

(Induction step.) Assume \( \sum_{j=0}^{N-1} d_{n+j} \leq M \) and show that \( \sum_{j=0}^{N-1} d_{n+1+j} \leq M \).
We have by the induction hypothesis:

\[ M \geq \sum_{j=1}^{N-1} d_{n+j} = \sum_{j=0}^{N-2} d_{n+1+j} + \delta_n \]

The processor causing delay \( d_{n+N} \) will have waited \((N-1)r + \delta_n\)
clock periods for the bit to return after initiation of its current
task. Since this task requires at most \( h \) clock periods to
complete, the processor can delay the bit for at most \( h \leq [(N-1)r + \delta_n]\)
clock periods before it completes its task and begins its input/output,
after which the bit proceeds to the next processor.
Hence, we have:  \( d_{n+N} \leq h \triangleq [(N-1)r + \delta_n] \) and:
\[
\begin{align*}
(\sum_{j=0}^{N-2} d_{n+1} + j) \leq \sum_{j=0}^{N-1} d_{n+1} + j + d_{n+N} \leq \delta_n + h \triangleq [(N-1)r + \delta_n] = M,
\end{align*}
\]
since \( \delta_n \leq M \).

This proves theorem (3-1). Note that the proof is independent of \( i \) and hence holds for all processor in the system.

Let \( d_{i-N+1}, d_{i-N+2}, \ldots, d_{i-1} \) represent a sequence of \( N-1 \) delays encountered by the control bit for \( i+N, N+1, \ldots \).

Define \( D_i \equiv \sum_{j=1}^{N-1} d_{i-j} \). Then the bit returns to the processor causing delay \( d_i \) exactly \( (N-1)r + D_i \) clock periods after the processor's current task began execution. The processor will not delay the bit at all if its current task has completed execution within these \( (N-1)r + D_i \) clock periods. It will delay the bit for \( k \) clock periods if the task is \( (N-1)r + D_i + k \) clock periods long, for \( k = 1, 2, \ldots, M-D_i \); this occurs with probability \( P'[\{(N-1)r + D_i + k] \).

The processor cannot delay the bit more than \( M - D_i \) clock periods according to theorem 3-1 above. Thus, we can define the conditional probabilities, \( pr[d_i = k | D_i] \equiv p(k, D_i) \), that a processor will delay the bit \( k \) clock periods given that the previous \( N-1 \) delays total \( D_i \) clock periods as follows:
\[
\begin{align*}
pr[d_i = 0 | D_i] & = P[(N-1)r + D_i] \\
pr[d_i = k | D_i] & = P'[\{(N-1)r + D_i + k] \\
& \text{for } k = 1, 2, \ldots, M-D_i \quad \text{if } D_i < M \\
pr[d_i = k | D_i] & = 0 \\
& \text{for } k = M-D_i, M-D_i + 1, \ldots, M
\end{align*}
\]
and

\[ \text{pr} \{ d_i = 0 | D_i \} = 1 \]

\[ \text{pr} \{ d_i = k | D_i \} = 0 \quad \text{if } D_i = M \]

\[ \text{for } k = 1, 2, \ldots, M \]

(3-1)

The probabilities are independent of the processor causing the delay.

Equations (3-1) indicate that knowledge of a sequence of

N-1 delays is sufficient to compute the probabilities for the N-th
delays's possible values. Hence, a Markov chain can be constructed
to model the sequence of delays encountered by the control bit.
The states of this chain are all possible N-1 length sequences of
delays, and the transition probabilities are equations (3-1).

A state is defined as an N-1 tuple of integers, \([k_1, k_2, \ldots, k_{N-1}]\),
such that:

1. \[ 0 \leq k_j \leq M \quad \text{for } j = 1, \ldots, N-1, \text{ and } \]
2. \[ \sum_{j=1}^{N-1} k_j \leq M . \]

Denote the set of states by S, with cardinality K.

Transitions from a state \([k_1, k_2, \ldots, k_{N-1}]\) to some state
\([k_2, k_3, \ldots, k_{N-1}, k]\) occur with probability \(\rho(k, \sum_{j=1}^{N-1} k_j)\).

All other possible transitions occur with zero probability.

For example, consider a system with 3 processors and task execution
times of 1 to 4 clock periods. Let \(r = 1\). Then we have \(M = 2\),
and a graphical representation of the Markov chain is shown in
figure 3-3. Note that systems with \(M = 0\) have only one state,
FIGURE 3-3: Markov Chain for an ORR System with $N=3$ and $h=4$
necessarily $[0, \ldots, 0]$. For these systems we have $h \leq (N-1)r$
which holds for all $N$ equal to $\left\lfloor \frac{h+r}{r} \right\rfloor$ or larger.

Markov chains defined in this manner are ergodic since they meet two sufficient conditions (see Parzen [PARZ 60], p. 140).
First, the chain has a state with a non-zero transition probability of returning to that state; state $[0,0,\ldots,0]$ meets this criterion since $\rho(0,0) = P[(N-1)r]$ if $M > 0$, i.e., $h > (N-1)r$, and
$\rho(0,0) = 1$ if $M = 0$. Second, all states of the chain "communicate," i.e. one can reach any state from any other state. This follows since the $[0,0,0,\ldots,0]$ state can be reached from any state, and any state can be reached from the $[0,0,\ldots,0]$ state. State $[0,0,\ldots,0]$ can be reached in at most $N-1$ steps from any state $[k_1,k_2,\ldots,k_{N-1}]$ by first moving to state $[k_2,\ldots,k_{N-1},0]$, and then state $[k_3,\ldots,k_{N-1},0,0]$, and so on; each of these transitions clearly has a non-zero transition probability. Similarly, any state $[k_1,k_2,\ldots,k_{N-1}]$ can be reached from state $[0,0,\ldots,0]$ within $N-1$ steps by first moving to state $[0,\ldots,0,k_1]$, and then $[0,\ldots,0,k_1,k_2]$, and so on; each of these transitions clearly has a non-zero transition probability, also. Hence, all states communicate.

Since the Markov chain is ergodic, the state probabilities can be computed, representing the probabilities of the system's being in one of the possible states after the system has reached statistical equilibrium. Let $\pi_{[k_1,k_2,\ldots,k_{N-1}]}$ represent the state probability for state $[k_1,k_2,\ldots,k_{N-1}]$. The probabilities are the solutions to the following system of linear equations:
a) K-1 equations of the form:

\[ \pi_{[k_1, k_2, \ldots, k_{N-1}]} = C \sum_{j=0}^{N-2} \pi_{[j, k_1, \ldots, k_{N-2}]} \cdot \rho(k_{N-1}; j + \sum_{\ell=1}^{N-2} k_\ell) \]

where \( C = M - \sum_{\ell=1}^{N-2} k_\ell \), and \( \sum_{k_1, k_2, \ldots, k_{N-1}} \).

b) one (constraint) equation of the form:

\[ \sum_{S_i} \pi_{[k_1, k_2, \ldots, k_{N-1}]} = 1 \]

From the state probabilities, the probability, \( \pi_i \), that the system is in any state with the N-1 delays summing to \( i \) can be defined:

\[ \pi_i \equiv \sum_{S_i} \pi_{[k_1, k_2, \ldots, k_{N-1}]} \]

where \( S_i \) is the set of states \( \pi_{[k_1, k_2, \ldots, k_{N-1}]} \) with \( \sum_{\ell=1}^{N-1} k_\ell = i \). Whenever the system enters any state \( \pi_{[k_1, \ldots, k_{N-1}]} \in S_i \), the control bit encounters a sequence of N-1 delays summing to \( i \) clock periods as it services N-1 processors. The average delay encountered in servicing all N-1 processors is then \( \sum_{i=0}^{M} i \cdot \pi_i \), and the average delay, \( t_f \), per processor is

\[ t_f = \frac{1}{N-1} \sum_{i=0}^{M} i \cdot \pi_i \]

since the delays are independent of the particular processors at which they occur. The throughput of the ORR system follows from the calculation of \( t_f \) according to equation (2-7). Note that systems with \( M=0 \) have \( TP = 1/r \) since \( t_f = 0 \). That is, systems
with $N \geq \left\lfloor \frac{h+r}{r} \right\rfloor$ have maximal throughput. Continuing the example of figure 3-3, we have the following system of equations:

$$
\pi_{00} = p(2) \pi_{00} + p(3) \pi_{10} + 1 \cdot \pi_{20}
$$

$$
\pi_{01} = p'(3) \pi_{00} + p'(4) \pi_{10}
$$

$$
\pi_{02} = p'(4) \pi_{00} + 1 \cdot \pi_{11}
$$

$$
\pi_{10} = p(3) \pi_{01} +
$$

$$
\pi_{11} = p'(4) \pi_{01} +
$$

$$
\pi_{20} = 1 \pi_{02}
$$

and the constraint equation:

$$
1 = \pi_{00} + \pi_{01} + \pi_{02} + \pi_{10} + \pi_{11} + \pi_{20}
$$

where $S = \{\pi_{00}, \pi_{01}, \pi_{02}, \pi_{10}, \pi_{11}, \pi_{20}\}$ and $K = |S| = 6$.

Solving this system of equations, we have:

$$
\pi_{00} = \frac{1-p'(4)}{c}
$$

$$
\pi_{01} = \pi_{10} = \frac{p'(3)}{c},
$$

$$
\pi_{02} = \pi_{20} = \frac{p'(4) - [p'(4)]^2}{c}, \text{ and}
$$

$$
\pi_{11} = \frac{p'(3) - p'(4)}{c}
$$

with $c = 1 + p'(4) + 2p'(3) - 2[p'(4)]^2 + p'(3)p'(4)$.

Thus, we have:

$$
\pi_{1} = \pi_{01} + \pi_{10} = \frac{2p'(3)}{c},
$$

$$
\pi_{2} = \pi_{02} + \pi_{11} + \pi_{20} = \frac{2p'(4) - 2[p'(4)]^2 + p'(3)p'(4)}{c}
$$
which leads to \( t_f = \frac{P'(3) + 2P'(4) - 2[P'(4)]^2 + P'(3)P'(4)}{c} \)

and \( TP = \frac{1}{t_f + 1} \).

If we use a uniform distribution for \( P \), i.e., if \( P'(i) = 0.25 \) for \( i = 1, 2, 3, \) and \( 4 \), then \( t_f = 0.407 \), and then system has a throughput of \( 0.7105 \). Simulation of this system yields a throughput of \( 0.7091 \) for the execution of 10000 tasks. This corresponds to an error of \( 0.120\% \) in the simulator's TP from the model.

Note that the simulator generated task execution times according to this distribution with a \( 0.2\% \) error in the average task execution time. The simulation of this model conforms very closely in other cases, as well.

The turnaround time for the ORR system follows from the above results and equation (2-8). It is composed of the three components used for the single processor system:

\( TT_i \), the average time that a task spends as an input packet,

\( TT_e \), the average time that a task spends in execution, and

\( TT_o \), the average time that a task spends as an output packet.

Since input packets enter the input register immediately after their predecessors exit, then they spend an average of \( t_f \) clock periods in the register waiting for a free processor and an additional \( r \) clock periods during input to the processor. Hence, we have \( TT_i = t_f + r \). The tasks spend \( p* \) clock periods
in execution; thus we have $TT_e = p^*$. The average time, $TT_W^*$, that
tasks wait for output (as output packets) is given in equation (2-8); to this is added $r$ clock periods for the output of the results from the processor:

$$TT_0 = TT_W + r = \frac{N}{TP} - p^*.$$

Together, the total turnaround time is:

$$TT = TT_i + TT_e + TT_0 = t_f + r + p^* + \frac{N}{TP} - p^* = \frac{N+1}{TP} = (N+1)(t_f + r). \quad (3-4)$$

Note that the derivation of equation (3-4) applies to all inter-
connection networks without buffering, as it is independent of the control mechanism employed.

An order of magnitude estimate for the number of states required
for the Markov chain can be computed.* Let $\binom{M}{N}$ states be required
for an ORR system with $N$ processors and parameter $M$. Each state
is an ordered $N-1$ tuple of the form $[k_1, k_2, \ldots, k_{N-1}]$ for positive integers $k_i$ constrained according to equations (3-2) and (3-3).

If $k_1 = 0$, then the number of possible $N-2$ tuples for $[k_2, k_3, \ldots, k_{N-1}]$ is $\binom{M}{N-1}$; similarly, if $k_1 = 1$, then exactly $\binom{M-1}{N-1}$ $N-2$ tuples are possible for $[k_2, k_3, \ldots, k_{N-1}]$. In general, if $k_1 = i$, then there exist $\binom{M-i}{N-1}$ $N-2$ tuples for $[k_2, k_3, \ldots, k_{N-1}]$.

*This calculation is due to Timothy Gonsalves and Bart Sinclair at Rice University.
Thus, $[M]_N$ can be defined recursively as:

$$[M]_N = [M]_{N-1} + [M-1]_{N-1} + \ldots + [M-i]_{N-1} + \ldots + [0]_{N-1}$$

$$= \sum_{i=0}^{M} [M-i]_{N-1} = \sum_{i_1=0}^{M} [i_1]_{N-1}$$

(3-5)

which each term in the summation corresponding to a distinct value for $k_1$ from 0 to $M$. Equation (3-5) can be rewritten as follows:

$$[M]_N = \sum_{i_1=0}^{M} \left( \sum_{i_2=0}^{M} \left[ i_2 \right]_{N-2} \right)$$

$$= \sum_{i_1=0}^{M} \sum_{i_2=0}^{M} \ldots \sum_{i_{N-3}=0}^{M} \left[ i_{N-2} \right]_{N-2}$$

(3-6)

where $[i_{N-2}]_{2} = i_{N-2} + 1$ trivially.

We have:

$$\sum_{i_{N-2}=0}^{i_{N-2}+1} (i_{N-2} + 1) = \sum_{i_{N-2}=1}^{i_{N-2}+1} i_{N-2}$$

Now, all summations of the form $\sum_{k=1}^{n} k^p$ can be expressed as
\[
\frac{n^{p+1}}{p+1} + A_1 n^p + A_2 n^{p-1} + \ldots \quad \text{for real constants } A
\]

[SPIV 67]. Hence, equation (3-6) can be rewritten as:

\[
\Gamma_{\binom{M}{N}} = \sum_{i_1=0}^M \sum_{i_2=0}^i \sum_{i_{N-3}}^{i_{N-4}} (-\frac{N-3}{2})^2 + O(1)
\]

\[
\Gamma_{\binom{M}{N}} = \frac{M^{N-1}}{(N-1)!} + O(N-2),
\]

(3-7)

where \(O(N-2)\) represents terms in \(M\) of order \(N-2\) or less. Using Sterling's approximation [KNUT 69] write \((N-1)!\) as \(\sqrt{2\pi(N-1)} \left(\frac{N-1}{e}\right)^{N-1}\) with relative error approximately \(1/12(N-1)\):

\[
\Gamma_{\binom{M}{N}} \approx \left(\frac{M \cdot e}{N-1}\right)^{N-1} \frac{1}{\sqrt{2\pi(N-1)}} + O(N-2).
\]
\[ M^e > (N-1) \left( 2 \pi (N-1) \right)^{-1/2(N-1)} \]

that is, when:

\[ h > (N-1) \left[ r + c(N) \right] > (N-1)(r+1) \]

where \( c(N) = \frac{1}{e} \left[ 2 \pi (N-1) \right]^{1/2(N-1)} \). For \( N=2 \), we have \( c(N) = 0.92214 \); for \( N = 10 \), we have \( c(N) = 0.4525 \); \( c(N) \) asymptotically approaches \( 1/e \) as \( N \to \infty \), and hence \( 0 < c(N) < 1 \) for all positive \( N \). We see that for many useful values of \( h \) and \( N \), the number of states required in the Markov chain and thus the amount of computation times required to solve for the performance measure become exceedingly large.

An approximation to the above analysis based on empirical evidence is presented here. It compares favorably with simulation results for the ORR system. Simulation, however, remains the most practical tool for accurate assessment of its performance. One observes from the solutions for several systems with small \( N \) and \( h \) that the following relation holds approximately:

\[ \pi[k,0,0,\ldots,0] \approx \pi[0,k,0,\ldots,0] \]

\[ \approx \pi[0,0,k,0,\ldots,0] \]

\[ \approx \pi[0,0,0,\ldots,0,k] \]

\[ \approx \frac{1}{c} \cdot p' [(N-1)r + k] , \]

for all \( k = 1,\ldots,M \) and some normalization constant \( c \).
Further, any state's N-2 tuple with exactly b non-zero entries \( k_1, k_2, \ldots, k_b \), appearing in that order from left to right, has state probability:

\[
\pi[k_1, k_2, \ldots, k_b, 0, \ldots, 0]
\]

\[
\approx \frac{1}{c} p' \left[ L + k_1 \right] \cdot p' \left[ L + k_1 + k_2 \right] \cdot \ldots \cdot p' \left[ L + k_1 + \ldots + k_b \right]
\]

\[
\approx \frac{1}{c} \prod_{j=1}^{b} p' \left[ L + \sum_{\ell=1}^{j} k_\ell \right]
\]

where \( L \equiv (N-1) \cdot r \) and \( b = 1, \ldots, M \). Note that there are exactly \( \binom{N-1}{b} \) ways in which the b non-zero entries can be placed in the N-1 available places so that their order is preserved since this is precisely analogous to the number of combinations of N-1 items taken b at a time. For example, if b = 2 and N = 5 we have the following \( \binom{4}{2} = 6 \) arrangements:

\[
[k_1, k_2, 0, 0], \quad [0, k_1, k_2, 0],
\]

\[
[k_1, 0, k_2, 0], \quad [0, k_1, 0, k_2],
\]

\[
[k_1, 0, 0, k_2], \quad [0, 0, k_1, k_2].
\]

The above observations follow naturally from the expected behavior of the ORR system. That is, the probability that a sequence of b delays \( k_1, k_2, \ldots, k_b \) is encountered by the control bit corresponds to the probability that b tasks of lengths \( L + k_1 \), \( L + k_1 + k_2 \), \ldots, \( L + k_1 + k_2 + \ldots + k_b \) occur in sequence. The effect of the intervening delays of length zero on the probability is unclear.
The probability for state \([0,0,...,0]\) can be calculated from equation (3-2) and the above approximation:

\[
\pi_{[0,0,...,0]} = \sum_{j=0}^{M} \pi_{[j,0,...,0]} \cdot \rho(0,j)
\]

\[
\approx \pi_{[0,0,...,0]} \cdot P(L)
\]

\[
+ \frac{1}{c} \sum_{j=1}^{M} P'(L+j) \cdot P(L+j)
\]

\[
\approx \frac{1}{c[1-P(L)]} \cdot \sum_{j=1}^{M} P'(L+j) P(L+j)
\]

The normalization factor \(c\) is just the sum of all the (approximate) state probabilities:

\[
C = \sum P'(L+\sum_{\ell=1}^{N-1} k_{\ell})
\]

\([k_1,...,k_{N-1}] \in S\)

where \(S\) is the set of all possible states. The delay probabilities; \(\pi_i\), are easily calculated from the state probabilities; substituting the above approximations, we have:

\[
\pi_i = \frac{1}{c} P'(L+i) \left\{ 1 + \sum_{b=2}^{i-1} \left( \sum_{\ell=1}^{N-1} k_{\ell} \right) \phi(i,b) \right\}
\]

where \(\frac{1}{c} P'(L+i) \phi(i,b)\) is the total state probability for all states \([k_1,...,k_{N-1}]\) with \(b\) non-zero entries and \(\sum_{\ell=1}^{N-1} k_{\ell} = i\) and:
\[ \phi (i, b) = \sum_{j_1=1}^{\Delta} p'(L+j_1) \cdot \sum_{j_2=1}^{\Delta+1-j_1} p'(L+j_1+j_2) \cdot \ldots \]
\[ \sum_{j_{k-1}=1}^{\Delta+k-1-k \sum (2j_1-1)} \ldots \sum_{j_{b-1}=1}^{i-1-k \sum (b-2)} \sum_{j_{b-1}=1}^{J_{b-1}} p'(L+j_1+\ldots+j_{b-2}+j_{b-1}) \]

where \( \Delta \equiv i - (b-1) \) and \( \Sigma(k) \equiv \sum_{m=1}^{k} j_m \). For example, suppose \( i = 5 \) and \( b = 3 \). Then, we have:

\[ \phi (5, 3) = p'(L+1) p'(L+2) \]
\[ + p'(L+1) p'(L+3) \]
\[ + p'(L+1) p'(L+4) \]
\[ + p'(L+2) p'(L+3) \]
\[ + p'(L+2) p'(L+4) \]
\[ + p'(L+3) p'(L+4) \]

Approximate values for throughput and turnaround time can be calculated from \( \pi_i \) as before. Let an approximation of "order k" denote a calculation using state probabilities with up to \( k \) non-zero entries.

Approximations of order 1, 2, and 3 for TP and TT are plotted versus the number of processor, \( N \), in figures 3-4 and 3-5 respectively. The task execution times are distributed uniformly with average 55.5 (clock periods) and the parameter \( r \) equals 1 clock period. The results of simulation runs executing 10,000 tasks for both the uniform and geometric distributions of task execution times are also shown. A line representing the ideal
FIGURE 3-4: Throughput Versus Number of Processors for the ORR System
r=1, p*=55.5, uniform distribution of task execution times

Throughput, TP

Number of Processors, N

○ = 1st order approximation
□ = 2nd order approximation
△ = 3rd order approximation
★ = simulation
FIGURE 3-5: Turnaround Time Versus Number of Processors for the ORR System
\( r=1, p^* = 55.5 \), uniform distribution of task execution times

- ○ = 1st order approximation
- □ = 2nd order approximation
- △ = 3rd order approximation
- ★ = simulation

Turnaround Time, TT

TT for single processor system

Number of Processors
N-fold improvement in throughput of an N-processor system over a single processor system and a line representing the turnaround time of a single processor system are included in figures 3-4 and 3-5 respectively for reference. The third order approximation for throughput is within $\approx 8.2\%$ of the simulation values with the maximal deviation occurring at $N \approx 30$. The latter appears to be nearly linear but with a slope that is 50.4\% less than the ideal N-fold improvement in throughput; this results in a throughput at $N = 55 \approx p^*$ which is 45.0\% less than the ideal value. The poor performance is explained by the following:

(a) the interconnection network delivers at most one input packet at a time to the processors since there is no buffering, and

(b) the control mechanism wastes clock periods waiting for a processor to complete its task when other processors are available to accept tasks.

The turnaround time, however, is always lower than the turnaround time of a single processor system and climbs linearly to this value.
3-3 Unordered Round Robin System

The unordered round robin (URR) system improves the poor throughput of the ORR system by making the control bit more readily available to the processors requiring service. This system keeps the control bit circulating continuously around the control loop at the rate of the system clock. When the bit locates a free processor, it remains at that processor for the duration of the transfer (r clock periods) and then moves on to the next processor during the final clock period of the transfer. This system does not in general preserve the order of the input stream of tasks in the output stream, hence the name "unordered" round-robin system. Consider again the example of the ORR system's behavior (figure 3-2). The behavior of the corresponding URR system with the same input stream is shown in figure 3-6. The control bit is initially at processor 1 and the three processors receive tasks with execution times 6, 4, and 8, as before. The control bit returns to processor 1 during clock period 6 and finds the processor busy in the execution of task 1. It thus moves on at clock period 7 to processor 2 which has completed its task by this time. Processor 2 receives an input packet during clock periods 8 and 9, and the bit circulates past processor 3 which is still in execution at clock period 10. The control bit returns at clock period 11 to processor 1 which has now completed task 1. The processor receives a new task, and so on. Note that the output packets for tasks 1 and 2 exit the system in the reverse order from which they arrived, illustrating that
Processor:

1  I/O  TASK 1  I/O  TASK 5  I/O  TASK 8

2  I/O  TASK 2  I/O  TASK 4  I/O  TASK 6  I/O  TASK 9

3  I/O  TASK 3  I/O  TASK 7

Time (in clock periods)

FIGURE 3-6: Illustration of the URR System's Behavior
this control scheme does not in general preserve the order of the packets.

A simple, exact analysis of the URR system is possible when only one clock period is required for the packet transfers (i.e. \( r=1 \)). In this case, the control bit visits each processor once in every \( N \) clock periods at intervals of exactly \( N \) clock periods. Since the input register contains an input packet at all times, then each processor has an opportunity to receive a new task (and deposit an output packet) at these times. And since the execution times for successive tasks are independent and identically distributed, then each processor has the same probability, \( \iota \), of actually taking a task from the input register when it is available. That is, as the control bit circulates around the control loop, the probability that a processor takes an input packet (and concurrently issues an output packet) remains the same. The throughput of the system then equals this probability according to the Law of Large Numbers. The turnaround time follows readily from \( \iota \) and equation 2-8. Hence, the analysis of the URR system with \( r=1 \) reduces to the determination of \( \iota \) for one processor in the system.

The behavior of a processor will be modeled similarly to the single processor system using the Markov chain of figure 3-7. The probability \( \iota \) will be shown to correspond to a state probability in the chain. The states shown in figure 3-7 are interpreted as follows. Initially, the processor enters the ready state at the arrival of the control bit. It remains in this state for
FIGURE 3-7: Markov Chain for the URR System with $r=1$
exactly N clock periods, during which it receives an input packet and executes the task for up to N-1 clock periods. If the task's execution time is less than N clock periods, then the processor is ready to receive a new task when the control bit returns after N clock periods. In this case, it re-enters the ready state, performs the input/output, and begins execution again. Otherwise, it enters the busy 1 state and continues execution of the task for up to N more clock periods. If the task's execution time is less than 2N clock periods, then the processor returns to the ready state upon the next return of the control bit; otherwise, it moves to the busy 2 state, and so on. In general, upon the i\textsuperscript{th} access to the input and output registers since the task began execution, the processor moves from the busy i-1 state to the ready state if the task's execution time is between (i-1)N and iN-1 clock periods; it moves instead to the busy i state if the task is at least iN clock periods in length. There are \( k \equiv \left\lfloor \frac{h}{N} \right\rfloor \) busy states in this chain since tasks are at most \( h \) clock periods long. Note that all state transitions take place at intervals of N clock periods, corresponding to successive arrivals of the control bit. Whenever the processor enters or re-enters the ready state, it receives a new task and deposits an output packet in the output register. Hence, the state probability that the processor is in the ready state equals the probability, \( l \), that the processor-and the system-receives/ completes a task.
The state probability for the \textit{ready} state and then the system's performance measures, are easily calculated using equation (2-3) once the transition probabilities are specified. These probabilities derive from the probability distribution, \( P(t) \), of the tasks' execution times. The probability, \( t_0 \), that the processor exits the \textit{ready} state simply equals the probability that the current task has an execution time exceeding \( N-1 \) clock periods:

\[
 t_0 = 1 - P(N-1) \quad . \tag{3-8}
\]

The probability, \( t_i \), that the processor moves from the \textit{busy} \( i \) state to the \textit{busy} \( i+1 \) state corresponds to the conditional probability that the task's execution time exceeds \( (i+1)N-1 \) clock periods given that it at least equals \( iN \) clock periods (otherwise, the processor would never have entered the \textit{busy} \( i \) state:

\[
 t_i = \frac{1-P[(i+1)N-1]}{1-P[iN-1]} \quad \text{for} \quad i = 1, 2, \ldots, k-1 \quad . \tag{3-9}
\]

Note that the probability that the processor moves from the \textit{busy} \( k \) state to the \textit{ready} state equals unity. Since the Markov chain in figure 3-7 is identical to the chain in figure 2-5 except for the number of busy states, then the solution for the state probability \( i \) is obtained from equation (2-3) by changing the index of summation from \( h \) to \( k \).
\[ l = \frac{1}{1 + \sum_{i=0}^{k-1} \sum_{j=0}^{i} t_{ij}} \]

\[ = \frac{1}{1 + \sum_{i=0}^{k-1} \{1 - P[(i+1)N-1]\}} \]  

(3-10)

In terms of the probability mass function, \( P'(t) \) corresponding to the distribution function \( P(t) \), we have:

\[ l = \frac{1}{1 + \sum_{i=0}^{k-1} h \sum_{j=(i+1)N}^{h} P'(j)} \]

\[ = \frac{1}{1 + \sum_{i=N}^{i=N} Q[i\frac{i}{N}] P'(i)} \]  

(3-11)

where \( Q[i\frac{i}{N}] \) denotes the quotient of the division of \( i \) and \( N \).

Note that if \( N=1 \), the throughput of the URR system conforms with the previously determined value for the single processor system. Also, for \( N > h + 1 \), we have \( TP = 1 \) since the summation vanishes (there are no busy states). The turnaround time follows immediately by substituting equation (3-8) into equation (3-4).

The throughput and turnaround time for the URR system with \( r = 1 \) are plotted versus the number of processors in figures 3-8 and 3-9 respectively. Task execution times are distributed uniformly and geometrically with average 55.5. The results of
FIGURE 3-9: Turnaround Time Versus Number of Processors for the URR System
\[ r=1, \ p^*=55.5 \]
the simulation of these systems conform to the calculated performance measures to within the accuracy of the simulator (measured by the error in the average task execution time from 55.5 clock periods). Comparison of figure 3-8 with figure 3-4 reveals that for uniformly distributed task execution times, the URR system exhibits approximately 25% higher throughput than the ORR system at \( N = 55.\bar{5} \); this improvement diminishes to zero as \( N \) approaches 1 and 110. The throughput and turnaround time for both systems are equal at these two extremes because at \( N = 1 \), both are single processor systems, and at \( N = 110 \), there are \( h + 1 \) processors so that the systems both attain the maximal throughput of unity.

An exact analysis of the URR system when the packet transfer time, \( r \), is greater than unity is possible. It closely resembles the analysis of the ORR system and hence requires considerably more complexity (and memory) than the above model for \( r = 1 \). The sequence of delays encountered by the control bit as it circulates around the control loop is modeled with a Markov chain. The throughput and turnaround time are calculated directly from the state probabilities of this chain.

Let \( d_1, d_2, \ldots, d_i, \ldots \) represent a sequence of such delays, where \( d_i \) is the \( i \)th delay encountered since the system initiated operation, for positive integers \( i = 1,2,3,\ldots \); this delay occurs at processor \( 1 + R[(i-1)/N] \equiv \phi(i) \). Unlike the delays defined in the ORR system's analysis, these delays represent the total time (in clock periods) that the control bit resides at the
processors. Each delay is then either $r$ or $l$ clock period(s) long, depending on whether the processor does or does not receive a new task. The parameter $M$ is defined as before:

$$M = h \cdot (N-1) \cdot r.$$ 

An obvious constraint on the sequence of delays can be stated as follows:

**Theorem 3-2:** If delay $d_i$, for positive integer $i$, equals $r$ and after $jN - 1$ subsequent delays, for all positive integers $j$, we have:

1. $d_{i+N} = d_{i+2N} = \cdots = d_{i+(j-1)N} = 1$ for $j \geq 2$, and
2. $\sum_{\ell=1}^{jN-1} d_{i+\ell} \geq h$,

then $d_{i+jN}$ must equal $r$.

This constraint says that if, after processor $\phi(d_i)$ receives a task, $h$ clock periods elapse without the processor receiving a new task, then the processor will receive a task at the next arrival of the control bit. It follows immediately from the fact the maximum task execution time is $h$ clock periods.

A second constraint on the sequence of delays serves as the basis for the construction of the Markov chain:

**Theorem 3-3:** In every sequence of $M = N(M-1) + N \equiv L$ delays, at least $N$ delays of length $r$, corresponding to all $N$ processors, are encountered. That is, each processor receives a new task at least once during each sequence of $L$ delays.
Proof: This observation can be proved by induction of \( i \), the number of delays encountered since the initiation of the system: (Basis step, \( i = 1 \).) There are \( N \) delays of length \( r \) in the first \( L \) delays since all processors are in the first \( L \) delays; all processors are initially idle and receive tasks when the control bit first reaches them. (Induction step.) Assume that at least \( N \) of the sequence of \( L \) delays \( d_1,d_{i+1}, \ldots, d_{i+L-1} \) equal \( r \) corresponding to all \( N \) processors and show that this is true of the \( L \) delays \( d_{i+1}, \ldots, d_{i+L} \).

Now, the sequence of \( L - 1 \) delays \( d_{i+1}, \ldots, d_{i+L-1} \) has either \( N \) or \( N-1 \) such delays according to the induction hypothesis. If it has all \( N \), then the theorem is proved trivially. If it has only \( N-1 \), then the delay \( d_{i} \) must equal \( r \) as it is the \( N^{th} \) delay for the sequence \( d_i, \ldots, d_{i+L-1} \). Also, none of the other delays \( d_{i+N}, d_{i+2N}, \ldots, d_{i+L-N} \) in the sequence which correspond to this processor equal \( r \). Hence, the task that begins execution in processor \( \phi(i) \) just after delay \( d_i \) does not complete before delay \( d_{i+L-N} \). However, the \( L-1 \) delays \( d_{i+1}, \ldots, d_{i+L-1} \) require at least \((N-1)r + L - 1 - (N-1)\) clock periods since at least \( N-1 \) of these delays are of length \( r \). We have:

\[
(N-1)r + L - 1 - (N-1) = (N-1)r + N \left\lfloor \frac{L}{N} \right\rfloor + N - 1 - (N-1)
\]

\[
= (N-1)r + N \left\lfloor \frac{h-(N-1)r}{N} \right\rfloor
\]

\[
= (N-1)r + N \left\lfloor \frac{h+r}{N} \right\rfloor - Nr
\]

\[
= N \left\lfloor \frac{h+r}{N} \right\rfloor - r \geq r
\]

.
Hence, the task must complete before delay \( d_{i+L} \), and the processor receives a new task at this time, i.e. \( d_{i+L} = r \). The \( L \) delays \( d_{i+1}, \ldots, d_{i+L} \) must then have at least \( N \) delays of length \( r \) corresponding to all \( N \) processors. Q.E.D.

The above constraints on the minimum number of delays of length \( r \) in a sequence of length \( L \) enables the calculation of the probability, \( \Pr[d_i = k \mid \vec{D}_i] = \rho(k, \vec{D}_i) \), that a delay \( d_i \) will equal \( k \) clock periods given the previous \( L - 1 \) delays \( d_{i-L+1}, \ldots, d_{i-1} \equiv \vec{D}_i \). If none of processor \( \phi(i) \)'s delays in the sequence \( \vec{D}_i \) equals \( r \), then \( d_i = r \) according to theorem 3-3. Otherwise, there exists some delay \( d_{i-nN} = r \equiv d_i^* \) for some positive integer \( n \) less than \( \frac{L}{N} \) such that all subsequent delays \( d_{i-jN} \) equal \( 1 \) for \( j = 1, \ldots, n-1 \); there are no subsequent delays if \( n = 1 \). The probability that \( d_i \) equals \( r \) is then the conditional probability that the task which began execution after delay \( d_{i-nN} \) completes before delay \( d_i \) given that it did not complete before delay \( d_{i-N} \). This probability equals:

\[
\Pr(\sum_{k=1}^{nN-1} d_{i-k} - \sum_{k=N+1}^{nN-1} d_{i-k})
\]

\[
= \frac{1 - \Pr(\sum_{k=N+1}^{nN-1} d_{i-k})}{1 - \Pr(\sum_{k=1}^{nN-1} d_{i-k})}
\]

if \( n > 1 \)

and \( \Pr(\sum_{k=1}^{N-1} d_{i-k}) \) if \( n = 1 \),

where \( \Pr \) is the probability distribution function of task execution times. The conditional probability is required to
take into account the fact that all delays corresponding to processor $\phi(i)$ in the sequence after delay $d_{i-nN}$, namely $d_{i-(n-1)N}, d_{i-(n-2)N}, \ldots, d_{i-N}$, equal 1, and hence the processor did not take a task at these opportunities. Note that the denominator of the above probability equals zero if

$$
\sum_{k=N+1}^{nN-1} d_{i-k} \geq h,
$$

but this implies that $d_{i-N}$ equals 1 according to theorem 3-2, which violates the assumption that $d_{i-N}$ equals one when $n > 1$. Hence we have:

$$
\sum_{k=N+1}^{nN-1} d_{i-k} < h,
$$

and that the conditional probability is well defined and non-zero. Thus, for a sequence of $L-1$ delays, $\hat{D}_i$, and delay $d_{i}^* = d_{i-nN}$ as defined above, we have:

$$
\rho(r, \hat{D}_i) = \begin{cases} 
\frac{\sum_{k=1}^{nN-1} d_{i-k}^{*}}{\sum_{k=N+1}^{nN-1} d_{i-k}^{*}} - \frac{\sum_{k=1}^{nN-1} d_{i-k}}{\sum_{k=N+1}^{nN-1} d_{i-k}} > 0 & \text{if } n > 1 \\
1 - \rho(\sum_{k=N+1}^{nN-1} d_{i-k}^{*}) & \text{if } n = 1 \\
1 & \text{if no delay } d_{i}^{*} \text{ exists, and}
\end{cases}
$$

$$
\rho(1, \hat{D}_i) = 1 - \rho(r, \hat{D}_i).
$$

(3-12)
Equations (3-12) indicate that knowledge of a sequence of \( L - 1 \) delays is sufficient to compute the probabilities for the \( L \)th delay's possible values. Hence, a Markov chain can be constructed to model the sequence of delays encountered by the control bit. The states of this chain are \( L - 1 \) length sequences of delays with transition probabilities derived from equation (3-12). These states are defined as the set of all \( L - 1 \) tuples \([k_1, k_2, \ldots, k_{L-1}] \equiv \vec{k}\) such that:

1. \( k_i \in \{1, r\} \) for \( i = 1, 2, \ldots, L - 1 \) and

2. \( L - 1 \) tuple \( \vec{k} \) is "reachable" from \( L - 1 \) tuple \([r, r, \ldots, r]\), as defined below.

The system of processors enters state \([r, r, \ldots, r]\) if it encounters a stream of \( L - 1 \) tasks of length \((N-1)r\) or less, in which case the processors are ready to receive new tasks whenever the control bit arrives; this event occurs with non-zero probability \( P[(N-1)r]^{L-1} \). The other states of the chain are found by enumerating all possible \( L - 1 \) length sequences of delays that can be reached with non-zero probability from this state. This is specified formally, as follows.

First we define the transition probabilities for the \( L - 1 \) tuples, and consequently, for the states of the chain. Let \( n \) be the smallest positive integer such that \( k_{L-n} = r \) for \( L - 1 \) tuple \( \vec{k} \). Then transition from \( L - 1 \) tuple \( \vec{k} = [k_1, k_2, \ldots, k_{L-1}] \) to \( L - 1 \) tuple \([k_2, k_3, \ldots, k_{L-1}, r]\) occurs with probability:
\[
\rho'(r, \vec{k}) = \begin{cases} 
\frac{nN-1}{\sum_{\ell=1}^{nN-1} k_{L-\ell}} - \sum_{\ell=N+1}^{nN-1} k_{L-\ell} & \text{if } n > 1, \\
1 - \sum_{\ell=N+1}^{nN-1} k_{L-\ell} & \text{if } n = 1, \\
1 & \text{if no such } n \text{ exists.}
\end{cases}
\]

Transition from \(L-1\) tuple \(\vec{k}\) to \(L-1\) tuple \([k_2, k_3, \ldots, k_{L-1}, 1]\) occurs with probability \(1 - \rho'(r, \vec{k}) = \rho'(1, \vec{k})\). All other transitions occur with zero probability. Then we have the following definition:

Definition: An \(L-1\) tuple \(\vec{a}\) is said to be reachable from \(L-1\) tuple \(\vec{a}\) if there exists a sequence of \(L-1\) tuples \(\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_i\) such that transitions:

1. from state \(\vec{a}\) to state \(\vec{x}_1\),
2. from state \(\vec{x}_j\) to state \(\vec{x}_{j+1}\) for \(j = 1, 2, \ldots, i-1\), and
3. from state \(\vec{x}_i\) to state \(\vec{b}\)

occur with probability \(p\), where \(0 < p \leq 1\). The states of the Markov chain are then the set \(S\), of all \(L-1\) tuples that are reachable from \(L-1\) tuple \([r, r, \ldots, r]\) with well defined, non-zero probabilities. These states thus represent all possible \(L-1\) length sequences of delays encountered by the control bit, since these sequences of delays produce well defined, non-zero transition probabilities. Hence, this Markov chain correctly models the delays encountered by the
control bit as it circulates past the processors.

As an example of this URR Markov chain, consider a system with \( r = 2, N = 2 \), and \( h = 4 \). Then we have \( M = 2 \) and \( L = 4 \); a graphical representation of this chain is shown in figure 3-10.

Note that the systems with \( M = 0 \) have \( n - 1 \) elements in each state. Hence only one state, with all elements equal to \( r \), can exist in this case. Also, we have \( \rho'(r,k) = 1 \) since no \( n \), as defined above, can exist. For these systems we have \( h \leq (N-1)r \) which holds for all \( N \geq \left\lfloor \frac{h+r}{r} \right\rfloor - 1 \).

The above Markov chain is ergodic since it meets the two sufficient conditions specified in Parzen [PARZ 60], p. 140. First, the chain has a state with a non-zero transition probability of reentering that state; the \([r,r,\ldots,r] \equiv r \) state meets this criterion since \( \rho'(r,r) = P [(N-1)r] \) if \( M > 0 \), i.e. if \( h > (N-1)r \), and \( \rho'(r,r) = 1 \) if \( M = 0 \). Second, all states of the "communicate", as defined for the ORR Markov chain. This follows since the \([r,r,\ldots,r]\) state can be reached from any state, and any state can be reached from the \([r,r,\ldots,r]\) state. State \([r,r,\ldots,r]\) can be reached in at most \( L-1 \) steps from any state \([k_1,k_2,\ldots,k_{L-1}]\) by first moving to state \([k_2,k_3,\ldots,k_{L-1},r]\), and then state \([k_3,k_4,\ldots,k_{L-1},r,r]\), and so on. Each of these transitions occurs with non-zero probability \( \rho'(r,k) \), since the states represent possible sequences of delays encountered by the control bit. Also any state \([k_1,k_2,\ldots,k_{L-1}]\) can be reached from state \([r,r,\ldots,r]\) according to the definition of the states of the chain. Hence, the Markov chain is ergodic.
Note illegal states:

\begin{verbatim}
[111]
[121]
\end{verbatim}

FIGURE 3-10: Markov Chain for the URR System
N=r=2, h=4
Since the Markov chain is ergodic, the state probabilities can be computed. Let $\pi_{[k_1, k_2, \ldots, k_{L-1}]}$ represent the state probability for state $[k_1, \ldots, k_{L-1}]$. These probabilities are solutions to a system of linear equations:

1. $K-1$ equations of the form:

$$
\pi_{[k_1, k_2, \ldots, k_{L-1}]} = \pi_{[r, k_1, \ldots, k_{L-2}]} \cdot \rho'(k_{L-1}, [r, k_1, \ldots, k_{L-2}]) + \pi_{[1, k_1, \ldots, k_{L-2}]} \cdot \rho'(k_{L-1}, [1, k_1, \ldots, k_{L-2}]),
$$

where $K$ is the cardinality of the set of states, $S$, and

2. one equation of the form:

$$
\sum_S \pi_{[k_1, k_2, \ldots, k_{L-1}]} = 1.
$$

From the state probabilities, the probabilities $\pi_1$ and $\pi_r$ that the system is in any state $\pi_{[k_1, k_2, \ldots, k_{L-1}]}$ with final delay $k_{L-1}$ equal to unity and $r$ clock periods respectively can be computed. Since the control bit is "symmetric" with respect to the processors, i.e. it favors no particular processor, then $\pi_1$ and $\pi_r$ respectively, represent the probability that the control bit will be delayed $1$ and $r$ clock periods at any processor. The throughput of the system is then:

$$
TP = \frac{\pi_r}{\pi_1 + r\pi_r}.
$$

(3-13)
since $\pi_r$ represents the average number of tasks completed per visit to a processor and $\pi_1 + r\pi_r$ represents the average duration of this visit. Note that with $M = 0$, we have $TP = \frac{1}{\tau}$ since the single state $[r, r, \ldots, r]$ causes $\pi_{L-1} \to 0$ to equal 1.

Continuing the example of figure 3-10, we have the following system of equations:

$$
\begin{align*}
\pi_{112} &= 1 \cdot \pi_{211} \\
\pi_{122} &= 1 \cdot \pi_{112} + 1 \cdot \pi_{212} \\
\pi_{211} &= (1-P_1)\pi_{221} \\
\pi_{212} &= P_1\pi_{221} \\
\pi_{221} &= (1-P_2)\pi_{122} + (1-P_2)\pi_{222} \\
\pi_{222} &= P_2\pi_{122} + P_2\pi_{222}
\end{align*}
$$

and the constraint equation:

$$
\pi_{112} + \pi_{122} + \pi_{211} + \pi_{212} + \pi_{221} + \pi_{222} = 1.
$$

where $S = \{\pi_{112}, \pi_{122}, \pi_{211}, \pi_{212}, \pi_{221}, \pi_{222}\}$ and $P \equiv P(i)$.

Solving this system of equations, we have:

$$
\begin{align*}
\pi_{112} &= \pi_{211} = \frac{(1-P_1)(1-P_2)}{c} \\
\pi_{122} &= \pi_{221} = \frac{1 - P_2}{c} \\
\pi_{212} &= \frac{P_1(1-P_2)}{c} \\
\pi_{222} &= \frac{P_2}{c}
\end{align*}
$$
with \( c = (3-P_1) (1-P_2) + 1 \). Thus, we have:

\[
\pi_1 = \pi_{211} + \pi_{221}
\]
\[
= \frac{(2-P_1)(1-P_2)}{c}
\]
\[
\pi_2 = \pi_{112} + \pi_{122} + \pi_{212} + \pi_{222} = \frac{2-P_2}{c}
\]

which leads to:

\[
TP = \frac{2 - P_2}{2 + (4-P_1)(1-P_2)}
\]

The turnaround time for the URR system is obtained by substituting equation (3-13) into equation (3-4).

The number of states required by this analysis may be very large, thus restricting its utility. It is interesting to note that fewer than \( L - 1 \) delays per state are required in some cases. First we show that knowledge of \( N \frac{h_i}{N-l} \) delays \( d_i, d_{i+1}, \ldots, d_{i+N-1} \) for positive integer \( i \) is sufficient to calculate the probability that the subsequent delay \( d_{i+N} \) will equal \( r \) or \( 1 \). As was observed earlier, in order to calculate this probability, the elapsed time since the current task began in processor \( \phi(d_i+N\frac{h_i}{N-l}) \) must be known.

Now, these \( N \frac{h_i}{N} \) delays are sufficient to calculate this elapsed time if the task began no earlier than with delay \( d_i \) (which corresponds to this processor); the task must have been no longer than \( N \frac{h_i}{N-l} - 1 < h \) clock periods for this to be possible.
However, if the task did not begin with any of the $\lfloor \frac{h}{N} \rfloor$ delays corresponding to processor $\phi(d_{i+N\lfloor \frac{h}{N} \rfloor})$ in this sequence, then it must have begun at the first delay, $d_{i-N}$, corresponding to this processor just preceding the sequence. This follows since the elapsed time between delays $d_{i-N}$ and $d_{i+N\lfloor \frac{h}{N} \rfloor}$ is at least $N\lfloor \frac{h}{N} \rfloor + N - 1 \geq h$ clock periods. In this case, delay $d_{i+N\lfloor \frac{h}{N} \rfloor}$ must equal $r$ with probability 1. Hence, only $N\lfloor \frac{h}{N} \rfloor$ delays are needed to construct a Markov chain modeling this system. However, it can be shown that $L - 1 < N\lfloor \frac{h}{N} \rfloor$ for $N, r \geq 2$ except when $r = 2$ and $R[\frac{h}{N}] = N - 1$, where we have:

$$L - 1 = N \lfloor \frac{h}{N} \rfloor + (N-1).$$

In these cases, only $N\lfloor \frac{h}{N} \rfloor$ delays are needed per state in the resulting Markov chain. This occurs, for example, when $N = r = 2$ and $h = 5$. 
3-4 The Left Priority System

The final control mechanism to be considered for interconnection networks without buffering is the left priority first (LPF) scheme commonly found in commercial systems. The system is shown in figure 3-11. Associated with every processor is an interface to the data busses and control lines. The control lines are connected to the control unit and serve to arbitrate the use of the busses as follows. One or more processors may simultaneously request use of the busses by asserting the bus request (BRQ) signal. When the busses are free, the global control unit responds by asserting bus grant (BGT). If processor 1 did not assert BRQ, it allows the assertion of BGT to propagate to processor 2, which does likewise. The processor closest to the control unit which asserted BRQ blocks the propagation of this signal to subsequent processors and thereby gains control of the busses. It asserts slave acknowledge (SACK) and lowers its BRQ. The control unit responds to SACK by removing BGT. When the processor completes its transfers, it lowers SACK which allows the control unit to assert BGT again in response to future or pending BRQ's. A timing diagram illustrates the coordination of control signals (figure 3-12). It should be noted that a processor can assert BRQ and capture BGT while the latter is propagating through the interfaces if it issues its BRQ before the upward transition of the incoming BGT. The processors thus have static priorities in the use of the bus determined by their relative places on the so-called "daisy chain" of the BGT control line.
FIGURE 3-12: Timing Diagram for the Control Signals in the Left Priority System

FIGURE 3-13: Typical Bus Grant (BGT) Capture Circuit
The behavior of the processor interfaces will be modeled by the \( N \) control flip-flops described in the introduction. The flip-flops are connected in the form of a shift register so that the control "bit" represents BGT in propagating at the rate of the system clock from the control unit to processors 1, 2, ... \( N \) in turn. In typical implementations, a processor's interface actually just delays BGT for the time required for the signal to clock the processor's BRQ into a flip-flop and block the further transmission of BGT (figure 3-13). Hence, the \( N \) flip-flop representation of BGT accurately models the delay (one clock period) required for the BGT to move from one processor to the next. We will ignore the skewing delays of the BRQ and SACK signals since they are at least an order of magnitude less than the corresponding processor's delay in receiving BGT; the effect of these delays becomes important when a large number of processors is used, as will be seen below. It is also assumed that the control unit offers negligible delay in the generation of BGT in response to BRQ and the de-assertion of SACK. These delays are shown graphically in figure 3-14. The processors remove SACK one clock period prior to the completion of their transfers so that BGT can be generated and arrive at processor 1 by the completion of the transfer. The behavior of the LPF model is illustrated in figure 3-15 where we have \( N = 3 \) and \( r = 2 \). Processor 1 gains control of the bus initially and receives task 1 at clock periods 2 and 3 after the control bit arrives during clock period 1. The control bit then moves to processor 2
Explanation of Delays:

\[ \delta_1 = \text{skewing delay for signal propagation}^* \]
\[ \delta_2 = \text{bus control unit circuit delay}^* \]
\[ \delta_3 = \text{BGT propagation delay} = \text{skewing delay} + \text{processor interface circuit delay} \]
\[ \delta_4 = \text{processor interface circuit delay}^* \]

*ignored in model

FIGURE 3-14: Signal Delays for Left Priority System
from the control unit at clock periods 3 and 4 and task 2 is transferred to the processor at times 5 and 6. Note that the control bit moves to processor 2 concurrently with the last clock period of the previous transfer. Processor 3 receives task 3 at clock periods 9 and 10 after the control bit propagates to it at times 6, 7, and 8. Task 1 is 6 clock periods long and completes during clock period 9; hence processor 1 receives the control bit during time 10 and receives task 4 at times 11 and 12, and so on. Note that processors 1, 2, and 3 receive the control bit with decreasing priority whenever two or more processors are concurrently available (e.g. initially).

Unlike the round-robin systems, the LPF system can never achieve the maximal throughput, \( TP = \frac{1}{r} \), for a single stream system. This constraint in performance is due to the delays inherent in the propagation of the control bit. Since tasks require at least one clock period to execute, the control bit can always reach processor 2 when processor 1 is busy. Even if tasks never exceeded one clock period, the maximal throughput would be:

\[
TP = \frac{2}{2r + 1} < \frac{1}{r},
\]

where one clock period is required for the BGT signal to reach processor 2. Hence, the throughput can never reach \( 1/r \) (although it approaches this value for large \( r \)).

The propagation delay of the control bit also restricts the number of processors that receive tasks:
Theorem 3-4: The maximal number of processors that ever receive tasks in the LPF system is the smallest integer $N_{\text{MAX}}$ such that:

$$N_{\text{MAX}} \geq \frac{1-2r + \sqrt{(2r+1)^2 + 8h}}{2}.$$

Proof: This can be proved by induction on the number of tasks transferred to the processors since the initiation of the system. Let $j_1, j_2, \ldots, j_i$ represent a sequence of positive integers with $j_i \in \{1, 2, \ldots, N_{\text{MAX}}\} \equiv S_p$ for positive integer $i$, corresponding to the sequence of packet transfers; element $j_i$ indicates that the $i^{th}$ task is input to processor $j_i$. Theorem 3-4 can then be proved by induction on the length, $i$, of this sequence, as follows. (Basis step: $i = 1, 2, 3, \ldots, N_{\text{MAX}}$.)

Clearly, the first $N_{\text{MAX}}$ tasks go processors $1, 2, \ldots, N_{\text{MAX}}$ or a subset thereof since the processors are initially idle.

(Induction step.) Assume that the first $i$ tasks go only to processors in $S_p$ for $i > N_{\text{MAX}}$, and show that the $i + 1$ st task does likewise.

Case I: $\{j_1, j_2, \ldots, j_i\} \subseteq S_p$. That is, at least one processor $x$ with $x \in S_p$ receives no task in the sequence of initiation $j_1, j_2, \ldots, j_i$. In this case, the processor corresponding to the minimal such $x$ will receive the $i + 1$ st task if the task is not taken by one of the processors in $\{j_1, j_2, \ldots, j_i\}$, and the theorem is trivially proved.
Case II: \( S_p \subseteq \{j_1, j_2, \ldots, j_i\} \). That is, all processors 1, 2, \ldots, \( N_{\text{max}} \) receive at least one task in the above sequence. Let \( j_k \) represent the element of this sequence with the largest index \( k \) such that \( S_p \subseteq \{j_k, j_{k+1}, \ldots, j_i\} \) and \( j_k \notin \{j_{k+1}, j_{k+2}, \ldots, j_i\} \). That is, starting with element \( j_i \) and moving backwards towards \( j_k \), \( j_k, j_{k+1}, \ldots, j_i \) is the shortest sub-sequence with \( S_p \subseteq \{j_k, j_{k+1}, \ldots, j_i\} \); clearly, such a subsequence, and thus an element \( j_k \), exists since \( S_p \subseteq \{j_1, j_2, \ldots, j_i\} \). We will show that the control bit cannot propagate past processor \( j_k \) in delivering the \( i + 1 \)st task to a processor. First, the minimal time required for the control bit to deliver the \( i-k \) tasks to processors \( j_{k+1}, j_{k+2}, \ldots, j_i \) is computed; the minimal time for this occurs when this subsequence contains the elements of \( S_p \setminus \{j_k\} \) with no repetitions. The time required for the control bit to deliver a task to some processor \( \ell \) is exactly \( \ell-1+r \) clock periods, where the bit propagates to the processor in \( \ell-1 \) clock periods and transfers the packets in \( r \) periods. Hence, the \( N_{\text{max}} - 1 \) processors are loaded with new tasks in a minimum of:

\[
\delta_1 = \sum_{\ell=1}^{N_{\text{max}}} (\ell-1+r) - (j_k-1+r)
\]

clock periods. The control bit requires an additional \( j_k-1 \) clock periods to reach processor \( j_k \) after the last processor is loaded. Hence, the control bit returns to processor \( j_k \) following the \( i \)th transfer at least:

\[
\delta_2 = \delta_1 + j_k - 1 = \frac{(N_{\text{max}}+1)(N_{\text{max}})}{2} + N_{\text{max}}(r-1) - r
\]
clock periods after the \(k^{th}\) transfer was completed. Now, if we have \(N_{\text{MAX}} \geq \frac{1 - 2r + \sqrt{(2r + 1)^2 + 8h}}{2}\), then it follows that \(\delta_2 \geq h\), and then the task received by processor \(j_k\) at the \(k^{th}\) transfer must be completed by the time the bit reaches this processor following the \(i^{th}\) transfer. If no processor \(x\) with \(x < j_k\) stops the control bit, then processor \(j_k\) will receive the \(i + 1\)st task; i.e. we have \(j_{i+1} \leq j_k \leq N_{\text{MAX}}\). Hence, no more than \(N_{\text{MAX}}\) processors can ever receive tasks. Q.E.D.

**Corollary 3-1:** All \(N_{\text{MAX}}\) processors are, in fact, used if a sequence of \(N_{\text{MAX}} - 1\) tasks of length \(h\) is encountered (which occurs with non-zero probability \([P'(h)]^{N_{\text{MAX}} - 1}\)).

**Proof:** If the first \(N_{\text{MAX}} - 1\) processors are available, then they will take these \(N_{\text{MAX}} - 1\) tasks (otherwise, a task will propagate to processor \(N_{\text{MAX}}\) at this time). Suppose these processors receive these tasks in arbitrary order \((k_1, k_2, \ldots, k_{N_{\text{MAX}} - 1})\) where each \(k_i\) is a unique element of the set \(\{1, 2, \ldots, N_{\text{MAX}} - 1\}\) for \(i = 1, 2, \ldots, N_{\text{MAX}} - 1\). Let \(t_1\) represent the time at which processor \(k_1\) begins execution of the first task of length \(h\); it completes this task at time \(t_1 + h\). The remaining processors, \(k_2, k_3, \ldots, k_{N_{\text{MAX}} - 1}\), require:

\[
\delta_3 = \sum_{\ell=2}^{N_{\text{MAX}} - 1} (k_\ell - 1 + r)
\]

clock periods to receive their tasks. We have:
\[ \delta_3 = \sum_{\ell=1}^{N_{\text{MAX}}-1} (k_{\ell} - 1 + r) - (k_1 - 1 + r) \]
\[ = \sum_{\ell=1}^{N_{\text{MAX}}-1} (\ell - 1 + r) - (k_1 - 1 + r) \]
\[ = \frac{N_{\text{MAX}}(N_{\text{MAX}}-1)}{2} + (N_{\text{MAX}}-2)(r-1) - k_1 \]  
(3-14)

since the \( k_{\ell}'s \) cover all of the integers from 1 to \( N_{\text{MAX}}-1 \) except for \( k_1 \). By theorem 3-4, it follows that:

\[ N_{\text{MAX}} < \frac{1-2r + \sqrt{(2r+1)^2 + 8h}}{2} + 1 \]  
(3-15)

Substituting equation (3-15) into equation (3-14), we have
\[ \delta_3 < h + 1 - k_1 \] . Now after delivering all of the length \( h \) tasks to processors 1,2,\ldots,N_{\text{MAX}}-1, processor \( N_{\text{MAX}} \) requests the control bit (assuming it is idle); the bit reaches processor \( \ell \) again at time \( t_1 + \delta_3 + \ell - 1 \) for \( \ell = 1,2,\ldots,N_{\text{MAX}}-1 \). But processor \( \ell \) is busy at least until time \( t_1 + \ell - 1 + r + h \) for \( \ell = k_1 \). We have \( t_1 + \ell - 1 + r + h > t_1 + \delta_3 + \ell - 1 \) since \( h + 1 - k_1 > \delta_3 \) and \( r > 1 - k_1 \) for all possible \( k_1 \); hence, no such processor \( \ell \) can stop the control bit. Processor \( k_1 \) cannot stop the bit either since it is busy until time \( t_1 + h \) and it receives the bit at time \( t_1 + \delta_3 + k_1 - 1 < t_1 + h \). The control bit propagates to processor \( N_{\text{MAX}} \) which receives the next task. Hence, all \( N_{\text{MAX}} \) processors can receive tasks with non-zero probability. Q.E.D.
In figure 3-16, the throughput and turnaround time for the
LPF system are plotted (using simulation) versus the number
of processors for both uniform and geometric distributions of
task execution times with $p^* = 55.5$ and $r = 1$. (The curves for
both distributions almost coincide.) The scale used conforms
with figures 3-4, 3-5, 3-8, and 3-9 in which the performance
measures of the ORR and URR systems are illustrated. In figure 3-17
the section of figure 3-16 for 1 to 20 processors is expanded;
the scaling of the ordinate remains the same. The throughput
increases nearly linearly and close to the line of $N$-fold improve-
ment for $N \leq 14$ and remains constant for all larger $N$ at a value
(0.169) considerably lower than the maximal throughput (1) for
a single stream system with $r = 1$. The turnaround time dips
initially and then rises with increasing $N$, in conformity with
equation (3-3) relating throughput and turnaround time. This
relationship fails to hold closely for $N > 14$. Theorem 3-4
predicts that no more than 15 processors ever receive tasks in
this system, and this is borne out by the simulation. (In fact,
no more than 14 processors ever receive tasks in the simulation.
This discrepancy is possibly explained by the very low probability
that the control bit propagates as far as the 15th processor.)
Only the first 15 processors contribute to the performance of the
system; and thus equation (3-3) is only valid for $N \leq 15$.
The performance of the LPF system is very poor relative to the
round-robin systems for these parameters due to the inefficiency
of the control protocol.
FIGURE 3-16: Throughput and Turnaround Time Versus Number of Processors for the Left Priority System
r=1, P*=55.5, uniform distribution
FIGURE 3-17: Throughput and Turnaround Time Versus Number of Processors for the Left Priority System
r=1, p*=55.5, uniform distribution
CHAPTER IV
INTERCONNECTION NETWORKS WITH BUFFERS

4-1 Introduction

In chapter three it was seen that interconnection networks without buffers achieve the maximal throughput allowed by the single stream only with a much larger number of processors than ideally required when task execution times are widely distributed.* This behavior is due to (1) the inefficiency with which the control mechanisms for these networks find available processors, and (2) the inability of these systems to deliver more than one task at a time to the processors when multiple processors are available to receive new tasks.

The interconnection networks described in this chapter alleviate these problems through the use of additional registers called **buffers** for the input and output packets. These registers match the very different characteristics of the single stream and the processors by allowing multiple tasks to be concurrently delivered to the processors. This effectively increases the average rate at which tasks are delivered to the processors and thus increases the throughput of the system. Unfortunately, the use of these buffers increases the time required for the individual tasks to reach processors, although this is offset in varying

* These systems achieve a throughput of 1/r with \( N_h = \left\lceil \frac{h+r}{r} \right\rceil \) processors where \( h \) is the largest task execution time and \( r \) is the packet transfer time. In chapter two it was shown that only \( N_p = \left\lceil \frac{p^*+r}{r} \right\rceil \) processors are ideally required to achieve this throughput, where \( p^* \) is the average task execution time. If \( h \gg p^* \), then \( N_h \gg N_p \).
amounts by the reduced time that the output packets wait to exit the processors. The net effect of these buffers on the average turnaround time depends on the ratio of the number of buffers, $R$, to the number of processors, $N$.

The first system discussed, called the *loop* system has one buffer per processor. This is the maximum amount of buffering used in the systems described in this thesis. The second system discussed, called the *hybrid* system, has multiple processors sharing one buffer. An interconnection network without buffers such as one of those described in chapter three can be used to connect the buffer to its processors; a slightly modified URR system is selected for analysis. Each buffer, its processors, and the interconnection network form one module of the complete system. As will be seen, the hybrid system represents a design methodology by which systems with various numbers of processors and buffers can be constructed to achieve a desired tradeoff in performance between throughput and turnaround time.
4-2 The Loop System

The loop system consists of \( N \) processors and \( L \)-bit buffers, one associated with each processor (see figure 4-1). The buffers are connected in a circular fashion along data paths \( W \) bits wide; each buffer is also connected to two processors. Input packets enter the system through buffer 1, which serves as the input register. They circulate continuously around the loop until a processor becomes available. Processor \( i \), for \( 1 \leq i \leq N - 1 \) removes an input packet from buffer \( i \) and concurrently places an output packet from the preceding task in buffer \( i + 1 \). Output packets exit the loop after reaching buffer \( N \), which is connected to the output register. Processor \( N \) receives its input packets from buffer \( N \), and it delivers its output packets directly to the output register when buffer \( N \) is not doing likewise. Output packets never circulate back to register 1 since we assume that the output register is always available. When input packets recirculate, they prohibit the input stream from placing a new input packet on the loop at those clock periods. All register-register and processor-register transfers of the packets require \( r \equiv \left\lfloor \frac{W}{N} \right\rfloor \) clock periods (denote this a transfer interval), and take place only at multiples of \( r \) clock periods.*

*Processors can complete tasks during the transfer intervals.
FIGURE 4-1: Block Diagram of the Loop System
An example of the behavior of this system is shown in figures 4-2 and 4-3 where \( N = 3 \) and \( r = 2 \). The system starts with an empty loop and with the input packet for task 1 in the input register (buffer 1). Task 1 is input to processor 1 at clock periods 1 and 2; task 2 reaches buffer 2 and is input to processor 2 at clock periods 5 and 6; task 3 reaches buffer 3 and is input to processor 3 at clock periods 9 and 10. Task 1 completes execution, and its output packet is placed in buffer 2 at clock periods 9 and 10. Concurrently with this transfer, processor 1 inputs task 5 from buffer 1. Task 1 finally exits the loop and reaches the output register at clock periods 13 and 14. Note that task 2's output from processor 2 to register 3 is delayed for two clock periods by the presence of an output packet in register 2 at clock period 11. Also note that task 3's output packet is transferred directly to the output register at clock periods 19 and 20. Task 4's input packet circulates in the loop through this short illustration without finding an available processor. Input packets can circulate for arbitrarily many clock periods in the loop with a finite but very small probability (see below). Finally, note that tasks exit the system in a different order than which they enter; the loop system does not in general preserve the order of the task stream.

The loop system is modular; a processor-register pair can be added or removed from the system without affecting its basic structure. A block diagram of a processor-buffer module illustrating the basic data and control paths is shown in figure 4-4.
FIGURE 4-2: Illustration of the Loop System's Behavior
<table>
<thead>
<tr>
<th>-   -   -</th>
<th>1   -   -</th>
<th>1   -   -</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>1   -</td>
<td>2   -</td>
<td>3   2   -</td>
</tr>
<tr>
<td>-------</td>
<td>----</td>
<td>-----</td>
</tr>
<tr>
<td>1   2   -</td>
<td>1*2   -</td>
<td>5   2*3</td>
</tr>
<tr>
<td>7</td>
<td>9</td>
<td>11</td>
</tr>
<tr>
<td>4   3   -</td>
<td>5   4   3</td>
<td>6   1   4</td>
</tr>
<tr>
<td>-------</td>
<td>----</td>
<td>-----</td>
</tr>
<tr>
<td>5   2*3</td>
<td>5   6   3</td>
<td>5   6   3</td>
</tr>
<tr>
<td>13</td>
<td>15</td>
<td>17</td>
</tr>
<tr>
<td>4   6  ^1</td>
<td>7   4  ^2</td>
<td>8   7   4</td>
</tr>
<tr>
<td>-------</td>
<td>----</td>
<td>-----</td>
</tr>
<tr>
<td>5   6   3*</td>
<td>5*6   7</td>
<td>9   6*7</td>
</tr>
<tr>
<td>19</td>
<td>21</td>
<td>23</td>
</tr>
<tr>
<td>4   8   7</td>
<td>9   4   8</td>
<td>8   5   4</td>
</tr>
</tbody>
</table>

Template:

\[ \text{time } \rightarrow t \quad p_1 \: p_2 \: p_3 \quad \text{+ processors} \]

\[ \text{b_1 \: b_2 \: b_3 \quad \text{+ buffers}} \]

where:

\[ t = \text{time of snapshot (i.e. just before clock period } t) \]
\[ p_i = \text{task in processor } i \quad (* \text{indicates that task is complete}) \]
\[ b_i = \text{task (i.e. packet) in buffer } i \quad (^\text{indicates an output packet}) \]

*Figure 4-3:* Behavior of the Loop System as Illustrated by System Snapshots
FIGURE 4-4: Loop System Modules
These modules are interfaced to the input and output streams, as indicated in the figure. Each processor allows the multiplexor to gate the contents of its buffer to the succeeding module at each transfer unless it is ready to receive a new task in which case it receives an input packet from the buffer and concurrently delivers an output packet to the succeeding module through the multiplexor. The output stream's interface issues control signal SYSTEM-READY whenever it finds buffer N empty or whenever it removes an output packet from the buffer; this allows the input stream's interface to route a new input packet into the input register (buffer 1). The control lines required to effect the transfers (e.g. clock signal for synchronous transfers) are not shown in figure 4-4. It is conceivable that the buffer and associated control circuitry for each processor-buffer module could be included on the same integrated circuit as the processor itself (in the case of a microprocessor), thereby reducing the cost of this additional hardware.

In the following analysis of the loop system, the performance measures are derived from the probability, $L_i$, that an input packet enters buffer $i$ at each transfer interval, for $1 \leq i \leq N$. This probability decreases for each successive buffer (i.e. for increasing $i$), since at each buffer a processor removes input packets from the loop. Since any packet must exit the buffer one clock period after it enters, $L_i$ equals the probability that buffer $i$ contains an input packet at each transition interval. Further, for $1 \leq i \leq N-1$, $L_{i+1}$ is the probability that
an input packet emerges from buffer \( i \) since all input packets that enter buffer \( i + 1 \) must emerge from buffer \( i \). Let \( \ell_{N+1} \) equal the probability that input packets emerge from buffer \( N \) and recirculate to buffer \( 1 \). It is easily observed that once all processors have initially received an input packet from the loop, then every buffer on the loop always contains either an input or an output packet. This follows since: (a) whenever an input packet is removed from the loop by a processor, it is replaced by an output packet, and (b) whenever the output stream's interface removes an output packet, the input stream's interface replaces it with a new input packet. Hence the probability that an output packet emerges from buffer \( i \) at each transfer interval equals \( 1 - \ell_{i+1} \). Since every output packet that emerges from buffer \( N \) exits the system, the throughput is just:

\[
TP = \frac{1 - \ell_{N+1}}{r}
\]  

(4-1)

(using the Law of Large Numbers). The turnaround time is derived from the throughput (see below). Approximate values for the \( \ell_i \)'s, and thus for the performance measures, are determined recursively by exploiting the modularity of the loop system, as follows.

Since each processor receives input packets only from its own buffer, the probability that a processor is busy (or idle) is dependent only upon (a) the probability that an input packet resides in this buffer, and (b) the distribution of task execution times. This observation derives from the fact that a processor
is never delayed in receiving a new task once the previous task is completed except by the absence of an input packet in its register. In particular, the time required to place an output packet on the loop does not delay the start of a new task since the removal of an input packet always provides a space for the output packet, and these operations are concurrent. Each processor-buffer module can then be modeled by a unique Markov chain (see figure 4-5) in which state changes occur once every \( r \) clock periods. The states of the chain are interpreted as follows. Initially, the processor resides in the \textit{idle} state wherein it is ready to receive an input packet. It moves to the \textit{busy 1} state at the first transfer interval after an input packet arrives in its buffer; for processor \( i \) this occurs with approximate probability \( \epsilon_i \) (see below). It resides in the \textit{busy 1} state for \( r \) clock periods, and returns to the \textit{idle} state if the task in execution completes during this time; otherwise it moves to the \textit{busy 2} state for another \( r \) clock periods. In general, after \( jr \) clock periods of execution, the processor will move from the \textit{busy j} state to the \textit{idle} state if the task's execution time is between \( (j-1)r + 1 \) and \( jr \) clock periods; otherwise it moves to the \textit{busy j + 1} state. There are then \( k = \left\lceil \frac{h}{r} \right\rceil \) busy states in this chain since tasks are at most \( h \) clock periods in length.

The transition probabilities for each chain are computed as follows. The transition probability, \( t_0 \), from the \textit{idle} state to \textit{busy state 1} in the chain for processor \( i \) is approximated
FIGURE 3-7: Markov Chain for the Loop System
by \( \ell_i \), the probability that an input packet enters buffer \( i \). Simulation shows that this is a good approximation, especially when \( \ell_i \) is close to unity, although it is slightly high (except when task execution times are geometrically distributed, in which case \( t_0 = \ell_i \) exactly). These results are intuitively reasonable since tasks enter a processor's buffer when the processor is busy and unable to accept them. The inaccuracy of this approximation is offset by a second approximation (below). Note that processor \( i \) re-enters the \textit{idle} state with probability \( 1 - \ell_i \) if no input packet is present.

The other transition probabilities are readily computed exactly from the probability distribution, \( P(t) \), of the tasks' execution times. The probability, \( t_j \), that the processor moves from the \textit{busy} \( j \) state to the \textit{busy} \( j+1 \) state corresponds to the conditional probability that the task's execution time exceeds \( jr \) clock periods given that it at least equals \( (j-1)r + 1 \) clock periods (otherwise the processor would never have reached the \textit{busy} \( j \) state):

\[
t_j = \frac{1 - P[jr]}{1 - P[(j-1)r]} \quad \text{for} \quad j = 1, 2, \ldots, k - 1 ;
\]

(4-2)

the processor returns instead to the \textit{idle} state with probability \( 1 - t_j \). Note that \( t_k = 1 \) since tasks never exceed \( h \) clock periods.

The \( \ell_i \)'s and hence the performance measures are computed approximately using the state probabilities of these Markov chains.
The probability, \( i \), that processor \( i \) is in the idle state is given by equation (2-3) where \( k \) is substituted for \( h \) in the index of summation, and \( t_0 = \ell_i \):

\[
\ell(i) = \frac{1}{1 + \ell_i \{1 + \sum_{j=1}^{k-1} [1 - P(jr)]\}}.
\]

In terms of the probability mass function, \( P'(t) \) corresponding to the distribution function \( P(t) \), we have:

\[
\ell(i) = \frac{1}{1 + \ell_i \{1 + \sum_{j=1}^{k-1} \sum_{m=jr+1}^{j} P'(m)\}}.
\]

\[
= \frac{1}{1 + \ell_i \{\sum_{j=1}^{h} (1 + Q[\frac{j-1}{r}]) P'(j)\}}. \tag{4-3}
\]

(Recall that \( Q[\frac{j}{r}] \) denotes the quotient of the division of \( j \) by \( r \).)

Let \( \ell^* = \sum_{j=1}^{h} (1 + Q[\frac{j-1}{r}]) P'(j) \), and let \( \beta(i) \equiv 1 - \ell(i) \) so that we have:

\[
\beta(i) = 1 - \frac{\ell_i}{1 + \ell_i \ell^*} = \frac{\ell_i \ell^*}{1 + \ell_i \ell^*}; \tag{4-4}
\]

\( \beta(i) \) is the (approximate) probability that the processor is busy and thus unable to receive a task. Note that \( \ell^* = \rho^* \) if \( r = 1 \). Now, probability \( \ell_1 \) equals unity since input packets enter buffer 1 at every opportunity, and output packets never recirculate to buffer 1 from buffer N. Hence, the probability that processor 1
is busy is just:

\[ \beta(1) = \frac{\ell^*}{1 + \ell^*} . \]

The probability that an input packet arrives at buffer 2 equals the joint probability that an input packet arrives at buffer 1 and finds processor 1 busy; we trivially have:

\[ \ell_2 = \beta(1) \]

since input packets are always arriving at buffer 1. The probability, \( \beta(2) \), that processor 2 is busy then equals:

\[ \beta(2) = \frac{\ell_2 \ell^*}{1 + \ell_2 \ell^*} = \frac{\beta(1) \ell^*}{1 + \beta(1) \ell^*} . \]

Similarly, the probability that an input packet arrives at buffer 3 is the joint probability that an input packet arrives at buffer 2 and finds processor 2 busy. If these two events are statistically independent, we have:

\[ \ell_3 = \ell_2 \cdot \beta(2) , \]

and in general:

\[ \ell_{i+1} = \ell_i \cdot \beta(i) = \frac{\ell_i \ell^*}{1 + \ell_i \ell^*} \quad \text{for } i = 1, 2, \ldots, N . \quad (4-5) \]

The arrival of an input packet to buffer \( i \) and processor \( i \)'s state are not statistically independent events according to equation (4-4) unless \( \ell_i = 1 \), but simulation indicates that equation (4-5) is a very good approximation. The dependence
occurs because each processor, upon completion of a task, waits for the arrival of the next input packet to its buffer before restarting execution. $\beta(i)$ is, however, relatively insensitive to changes in $\ell_i$ until $\ell_i\ell^*$ approaches unity, i.e. until $\ell_i$ approaches zero, if $\ell^*$ is large as illustrated in figure 4-6 for $\ell^* = 55.5$. Simulation reveals that the values obtained for $\ell_{i+1}$ using this approximation are slightly low. The values for $\beta(i)$ are, however, slightly high, and thus these two approximations tend to offset one another, as borne out by simulation. The approximate throughput of the loop system follows immediately from equation (4-1) by solving recursively for $\ell_{N+1}$ using equation (4-5). Note that if $r = 1$, then we have $\ell^* = p^*$, and in this approximate solution, the throughput depends only on the average task execution time.

The turnaround time for the loop system can be calculated from the throughput as follows. It is subdivided into three components:

- $TT_\ell$, the average time that a task spends on the loop as both an input and output packet, including the time required for input to a processor and output from the loop to the output register (for processors 1, 2, ..., N-1 only),

- $TT_e$, the average time that a task spends in execution within a processor,

- $TT_w$, the average time that a task spends waiting to be output from a processor, and
\( \mathbb{T}_0 \), the average time required to output an output packet from a processor to either the loop (for processors 1, 2, ..., N-1) or directly to the output register (for processor N).

The first component is a multiple of \( N_r \) clock periods. Consider processor \( i \), for \( i = 1, 2, ..., N-1 \); and input packet reaching processor \( i \) will require \((i-1)r\) clock periods to reach buffer \( i \) from buffer 1. It may circulate around the loop \( k \) times before the processor takes it, requiring \( kN_r \) clock periods. The input packet enters processor \( i \) in \( r \) clock periods, and after the task is executed, the output packet appears in buffer \( i+1 \). Then, \([N-(i+1)]r\) clock periods are required for this packet to reach buffer \( N \) and \( r \) additional clock periods to reach the output register. Adding up these delays we have:

\[
\mathbb{T}_\ell = (i-1)r + kN_r + r + [N-(i+1)]r + r = (k+1)N_r.
\]

(4-6)

If \( i = N \), then the output packet does not reach the output register from the loop, and we have:

\[
\mathbb{T}_\ell = (N-1)r + kN_r + r = (k+1)N_r,
\]

as before. Now, if the loop system is active for \( T \) clock periods, then in the limit as \( T \) grows without bound, \( \ell_{N+1} \). \((T/r)\) input packets recirculate* and TP-T tasks are completed; hence the average number, \( k \), of recirculations per task is:

\( \ell_{N+1} \) is the probability that an input packet exits buffer \( N \) at each transfer interval of \( r \) clock periods.
\[
\lim_{T \to \infty} \frac{1}{r} \frac{\ell_{N+1}}{TP} \frac{T}{T_0} = \frac{\ell_{N+1}}{r \cdot TP} = k.
\]

Substituting into equation (4-6), we have:

\[
TT = (\frac{\ell_{N+1}}{r \cdot TP} + 1) \frac{Nr}{1 - \frac{\ell_{N+1}}{r}} = \frac{1}{1 - \frac{\ell_{N+1}}{r}} \frac{Nr}{TP}
\]

The second component, \(TT_e\), is trivially equal to \(p^*\). The third component, \(TT_w\), is given by equation (2-8):

\[
TT_w = \frac{N}{TP} - (p^* + r)
\]

The final component, \(TT_0\), is trivially equal to \(r\) clock periods. Note that processor \(n\) transfers its output packet directly to the output register, rather than to a loop buffer. The total turnaround time then equals:

\[
TT = TT_i + TT_e + TT_w + TT_0
\]

\[
= \frac{N}{TP} + p^* + \frac{N}{TP} - (p^* + r) + r
\]

\[
= \frac{2N}{TP}
\]

(4-7)

A solution for the recurrence relation in equation (4-5) can be constructed as follows. Let \(\mu_i = \frac{1}{\ell_i}\) and \(c = 1/\ell^*\). Substituting into equation (4-5), we have:

\[
\mu_{i+1} = \mu_i + c(1 + \mu_i) \quad \text{for} \quad i = 2, 3, \ldots, N,
\]

(4-8)

with \(\mu_2 = 1 + c\). (Note that \(\mu_1 = 1/\ell_1 = 1\).) We have the following result.
Theorem 4-1: \( \frac{1}{1+c} \mu_i \) is a polynomial of order \( 2^{i-2} - 1 \) in \( c(1+c) \equiv z \); i.e.

\[
\frac{1}{1+c} \mu_i = \sum_{j=0}^{2^{i-2}-1} k_{ij} z^j
\]  \((4-9)\)

where the \( k_{ij} \)'s are constants.

Proof: The proof is by induction on \( i \).

(Basis step: \( i = 2 \)) We have \( \frac{1}{1+c} \mu_2 = 1 \) which is a polynomial of order zero in \( z \). (Induction step) Assume that \( \frac{1}{1+c} \mu_i \)
is a polynomial of order \( 2^{i-2} - 1 \) in \( z \) and show that \( \frac{1}{1+c} \mu_{i+1} \) is a polynomial of order \( 2^{i-1} - 1 \) in \( z \). Equation (4-8) can be rewritten as:

\[
\frac{1}{1+c} \mu_{i+1} = \frac{1}{1+c} \mu_i + \underbrace{c(1+c)}_{I} \left( \frac{1}{1+c} \mu_i \right)^2 + \underbrace{1}_{II}.
\]  \((4-10)\)

Now, the highest exponent of \( z \) in the above equation occurs in term II and equals \( 1 + 2(2^{i-2} - 1) = 2^{i-1} - 1 \); thus \( \frac{1}{1+c} \mu_{i+1} \)
is a polynomial of order \( 2^{i-1} - 1 \) in \( z \). Q.E.D.

The throughput follows immediately from the calculation of \( \mu_i \):

\[
TP = 1 - \frac{1}{\mu_i}.
\]  \((4-11)\)

In order to calculate \( \mu_i \), the constants, \( k_{ij} \), in equation (4-9) must be determined. Substituting equation (4-9) into equation (4-10), we have:
\[ \sum_{j_1=0}^{2^{i-2} - 1} \sum_{k_{i+1,j_1} z_1} = \sum_{j_2=0}^{2^{i-2} - 1} z_2 + \left( \sum_{j_3=0}^{2^{i-2} - 1} k_{i,j_3} z_3 \right) \left( \sum_{j_4=0}^{2^{i-2} - 1} k_{i,j_4} z_4 \right). \] (4-12)

Each coefficient \( k_{i+1,j} \) can be expressed recursively by summing all coefficients, \( k_{i,j} \), of \( z^j \) terms on the right hand side of equation (4-12):

\[
k_{i+1,j} = \begin{cases} 
  k_{i,j} & \text{for } j = 0, \\
  k_{i,j} + \sum_{\ell=0}^{j-1} k_{i,\ell} k_{i,j-1-\ell} & \text{for } j = 1,2,\ldots,2^{i-2} - 1 \\
  2^{i-2} \sum_{\ell=L_1}^{2^{i-2} - 1} k_{i,\ell} k_{i,2^{i-2} - 1 + L_1 - \ell} & \text{for } j = 2^{i-2},2^{i-2} + 1,\ldots,2^{i-1} - 1,
\end{cases}
\]

for \( i = 2,3,\ldots,N \) and \( j = 1,2,\ldots,2^{i-1} - 1 \),

(4-13)

where \( L_1 = R \lfloor j/2^{i-2} \rfloor \) and \( k_{2,0} = 1 \). Note that for \( j \geq 2^{i-2} \) the first summation on the right hand side of equation (4-12) does not contribute any terms to \( k_{i+1,j} \). For small \( j \), closed form solution for \( k_{i,j} \) follow readily from equation (4-13):

\[
k_{i,0} = 1, \\
k_{i,1} = i-2, \\
k_{i,2} = (i-2)(i-3) \leq (i-2)^2,
\]
\[ k_{i,3} = \frac{1}{2} (i-2)(i-3)(2i-7) \leq (i-2)^3 , \]
\[ k_{i,4} = \frac{1}{2} (i-2)(i-3)(2i-7)(i-4) \frac{10+6(i-5)}{9+6(i-5)} \leq (i-2)^4 . \]

For larger \( j \), solutions for \( k_{i,j} \) become exceedingly difficult to generate. However, the following bound on \( k_{i,j} \) can be proved by double induction on index variables \( i \) and \( j \):

**Theorem 4-2:** \[ k_{i,j} \leq (i-2)^j \] (4-14)

(See the appendix for proof.) Plugging this bound into equation (4-9), we have:

\[ \mu_i \leq (1+c) \sum_{j=0}^{2^{i-2}-1} [(i-2)z]^j . \] (4-15)

For \((i-2)z \ll 1\), the magnitude of succeeding terms in equation (4-15) decreases geometrically. Assuming that the low order \( k_{i,j} \)'s are reasonably close to the bound and hence, much more significant than the high order terms (this is the case for the first four \( k \)'s), then the series can effectively be approximated by the truncation of high order terms. Since

\[ TP = \frac{1}{\mu_i} \left( 1 - \frac{1}{\mu_{N+1}} \right) \] according to equation (4-1), then this approximation to the throughput of a loop system with \( N \) processors is valid when \( N \ll 2 + \frac{\ell^*2}{1+\ell^*} \). Denote \( \sum_{j=0}^{n} k_{i,j} z^j \) an "\( n \)th order" approximation to \( \mu_i \). As an example, consider a system with \( p^* = 55.5 \), \( r = 1 \), and \( N = 10 \). We have \( \ell^* = 55.5 \) and \( 2 + \frac{\ell^*2}{1+\ell^*} = 56.518 \). Table 1 gives first through fourth order approximations to \( \mu_i \) and TP. The fourth order approximation
<table>
<thead>
<tr>
<th>j</th>
<th>$k_{11,j}$</th>
<th>$k_{11,j} \cdot z^j$</th>
<th>$\sum_{j=0}^{j} k_{11,j} e^{z^j}$</th>
<th>TP, jth order approximatin</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>9</td>
<td>.165084003</td>
<td>1.165084003</td>
<td>0.156884068</td>
</tr>
<tr>
<td>2</td>
<td>72</td>
<td>.24224647</td>
<td>1.189308650</td>
<td>0.174057226</td>
</tr>
<tr>
<td>3</td>
<td>540</td>
<td>.003332585</td>
<td>1.192641235</td>
<td>0.176365149</td>
</tr>
<tr>
<td>4</td>
<td>3864</td>
<td>.000437408</td>
<td>1.193078643</td>
<td>0.176667111</td>
</tr>
</tbody>
</table>

$c = .018018018$

$z = .018342667$

TP from simulation = .180180180

(1.949753074 % error)

TP from recursion = .1764005665

(0.151101839 % error)

TABLE 4-1: Approximations to the throughput of a Loop System with $p^* = 55.5$ and $N=10$
is within 0.152 percent of the throughput obtained in the recursive analysis.

In figures 4-7 and 4-8, the throughput and turnaround time are respectively plotted versus the number of processors for a loop system with $p^* = 55.5$ and $r = 1$, using values obtained from the recursive model, the fourth order approximation to this model, and simulation with both uniform and geometric task execution times. The approximation diverges from the recursive model as $N$ approaches $2 + \frac{p^*_2}{1 + p^*_2}$ but converges again for large $N$. The latter effect is due to the fact that $u_i$ grows rapidly with $N$ when the terms of the summation in equation (4-15) are larger than 1, and the throughput is inversely proportional to $u_i$; hence TP quickly approaches its limiting value of unity for these values of $N$. The recursive model agrees very well with simulation. Its maximum error occurs for $N \approx p^*$, at which point $\xi_{N+1}$ approaches zero and the approximations used in the model become inaccurate. The relative behavior of the loop system to the systems discussed in chapter three is discussed below.

It is clearly more practical to compute TP recursively using equations (4-5) and (4-1) than to use the above solution to equation (4-5), based on the relative computation times. The recursive calculation has the added advantage that the throughput for all systems with one through $N$ processors is obtained in the calculation of TP for the system with $N$ processors.

The above model reduces the state space from $N^{k+1}$ to $N(k+1)$ states where $k = \left\lfloor \frac{h}{r} \right\rfloor$ by decomposing the system into $N$ separate
FIGURE 4.7: Throughput Versus Number of Processors for the Loop System

Throughput, $TP$

Number of processors, $N$

- recursive model
- 4th order approximation to recursive model
- simulation
FIGURE 4-8: Turnaround Time Versus Number of Processors
for the Loop System, r=1, p=55.5, uniform and geometric distributions

4th order approximation to recursive model

Simulation

recursive model
Markov chains and introducing an approximation. We outline here an exact model of the loop system; this model illustrates the economy of the above techniques and the nature of the approximation used. Recall that the processors of the system are individually either idle or in one of \( k \) busy states. Let the \( N \)-tuple \( X_1X_2\ldots X_N \equiv \hat{X} \) represents a state of the system, where \( X_i \in \{ I, B_1, B_2, \ldots, B_k \} \) for \( i = 1, 2, \ldots, N \). The system is defined to be in state \( \hat{X} \) at time \( t + Nr \) if processor \( i \) is (was) in state \( X_i \) at time \( t + ir \), for positive integer \( t \). Note that there are \( N^{k+1} \) states of the system. Transitions from state \( X_1^1X_2^1\ldots X_N^1 \equiv \hat{X}_1 \) to state \( X_1^2X_2^2\ldots X_N^2 \equiv \hat{X}_2 \) are governed by the following restrictions based on the actual behavior of the loop system:

1. if \( X_i^1 = I \) then \( X_i^2 = B_1 \) since buffer 1 always contains an input packet;

2. if \( X_i^1 = I \) for \( 2 \leq i \leq N \), then \( X_i^2 = B_1 \) if and only if \( X_j \neq I \) for \( j = 1, 2, \ldots, i-1 \) since an input packet is in buffer \( i \) at time \( t + ir \) if and only if the input packet that is in buffer 1 at time \( t + r \) finds processor \( j \) busy when it reaches buffer \( j \) at time \( t + jr \); otherwise \( X_i^2 = I \);

3. if \( X_i^1 = B_k \), then \( X_i^2 = I \) since tasks are no more than \( h \) clock periods long;

4. if \( X_i^1 = B_j \) for \( 1 \leq j < k \), then \( X_i^2 = B_{j+1} \) with probability \( t_j \) according to equation (4-2), and \( X_i^2 = I \) with probability \( 1 - t_j \) for the same reasons as in the recursive model.
Since the tasks in the processor are independent, then the transition probability from state $X^1_i$ to state $X^2_i$ is just
\[ \tilde{X} = x_1 \cdot x_2 \cdot \ldots \cdot x_N, \]
where $x_i$ is the probability, as given above, that processor $i$ moves from state $X^1_i$ to state $X^2_i$, for $i = 1, 2, \ldots, N$. Note that all states $\tilde{X}$ such that $X^1_i = X^1_{i+1} \not= I$ for $i = 1, 2, \ldots, N-1$ are forbidden by rule 2 even though the transition probability into these states is well defined for $X_i \not= B_1$.

This rule prevents the system from entering any state $\tilde{X}^2$ with $X^2_i = X^2_{i+1} = B_1$ for $i = 1, 2, \ldots, N-1$ since this state is entered from state $\tilde{X}^1$ with $X^1_i = X^1_{i+1} = I$, and this transition for $X^1_{i+1}$ is forbidden if $X^1_i = I$. All succeeding states $\tilde{X}$ with $X_i^1 = X^1_{i+1} = B_j$ for $j = 2, 3, \ldots, k$ can only be reached from state $\tilde{X}^2$ and hence are never entered either. The state set for this chain is then all $N$-tuples $\tilde{X}$ such that $X_i \in \{I, B_1, B_2, \ldots, B_k\}$ for $i = 1, 2, \ldots, N$, and if $X_i^1 = X^1_{i+1}$ then $X^1_i = I$ for all $i = 1, 2, \ldots, N-1$. As an example, let $N = 2$, $h = 2$, and $r = 1$.

The chain for this trivial system is shown in figure 4-9; note that states $B_1 B_1$ and $B_2 B_2$ are missing.

The throughput of the loop system is easily calculated from the chain's state probabilities. Let $\pi_{\tilde{X}}$ represent the state probability for state $\tilde{X}$, and let $S_B$ represent the set of all states $\tilde{X}(=X_1^1 X_2^1 \ldots X_N^1)$ in which all processors are in some busy state $B_j$. That is, $S_B = \{ \tilde{X} | X_i^1 \not= I \text{ for } i = 1, 2, \ldots, N \}$. Whenever the system is in a state $\tilde{X} \in S_B$ at time $t+N$, then the input packet in buffer 1 at time $t+1$ returns to this buffer at time $t+N+1$ since all of the processors are busy at
FIGURE 4-9: Markov Chain for the Complete Model of the Loop System
N=2, h=2, r=1
times $t + i$ for $i = 1, 2, \ldots, N$. Thus, the probability that the system is in some state in which an input packet recirculates to buffer 1 is just

$$\sum_{X \in S_B} \pi_X = \ell_{N+1},$$

and the throughput follows immediately from equation (4-1). In the chain of figure 4-9, we have $S_B = \{B_1B_2, B_2B_1\}$ and:

$$\ell_3 = \pi_{B_1B_2} + \pi_{B_2B_1} = \frac{p(2)}{2p(2)+1},$$

where $p(2)$ is the probability that a task is two clock periods long. We can compare this result to the recursive model. We have $\ell^* = p^* = 1 + p(2)$ and:

$$\ell_2 = \frac{\ell^*}{1+\ell^*},$$

$$\ell_3 = \frac{\ell_2 \ell^*}{1+\ell_2 \ell^*} = \frac{[1+p(2)]^2}{2+p(1) = [1+p(2)]^2}. $$

When $p(2) = 0.5$, then $\ell_3 = 0.25$ according to the exact model, and $\ell_3 = 0.2842$ according to the recursive model; simulation of this system yields $\ell_3 = 0.2491$, which is very close to the exact model.

The simple example of figure 4-9 helps to explain the approximation made in the recursive loop model. When $p(2) = 0$, then the exact model predicts that $\ell_3 = 0$ (verified by simulation), whereas the recursive model predicts that $\ell_3 = 1/3$. Now, input
packets arrive at buffer 2 at every other clock period since processor 1 executes a task within one clock period \((p(2) = 0)\). Hence, processor 2 can always execute the task in its input buffer before the next one arrives. The recursive model only takes into account the average arrival rate of input packets to buffer 2 (i.e., it correctly predicts that \(l_2 = 0.5\)), not the distribution of their interarrival times. Hence, it incorrectly predicts that \(l_3\) is non-zero, which is true only if two or more input packets stream into buffer 2 at consecutive clock periods. In general, any exact model of the loop system must take into account the distribution of packet interarrival times at each buffer. However, the recursive model gives good approximate values for TP when \(\ell^{*}l_i \gg 1\) since (a) when \(\ell^{*}\) is very large, multiple packets pass through buffer \(i\) during the execution of our task, and thus the distribution function for their interarrival times appears to have a diminished effect on the performance measures (especially when \(l_i\) is close to unity), and (b) when \(l_i\) is close to unity, then the approximations made in the recursive analysis are very good. Figures 3-7 and 3-8 show that the recursive analysis and simulation give very similar results for systems with \(p^{*} = 55.5\) and \(r = 1\).

The performance of the loop system differs significantly from the performance of the systems described in chapter three in which there are no buffers in the interconnection networks. As indicated by figure 4-7 where \(p^{*} = 55.5\) and \(r = 1\), the loop system's throughput is very close to the ideal \(N\)-fold
improvement for an N processor system over that for the single processor system, and thus it is much better than the ORR, URR, and LPF systems (see figures 3-4, 3-8, and 3-16). Figure 4-8 indicates that the loop system's turnaround time is, unfortunately, much higher than the other systems (see figures 3-5, 3-9, and 3-17) due to the use of the buffers. For \( N < p^* \) the turnaround time remains fairly constant at approximately the value for the single processor system; this is predicted by equation (4-7):

\[
TT = \frac{2N}{TP} \approx \frac{2N}{N \cdot TP_1} = TT_1
\]

where \( TP_1 \) and \( TT \) are respectively the throughput and turnaround time of the single processor system, and \( TP \approx N \cdot TP_1 \).

For \( N > p^* \), the turnaround time rises asymptotically to the line \( TT = 2N \) since \( TP \approx 1 \) for these values of \( N \). For \( N \approx p^* \), this system achieves the maximal throughput allowed by the single stream with a turnaround time nearly equal that for the single processor system.

The loop system represents an extreme in the performance of a load-distributed multiprocessor connected to a single stream of tasks. Unlike systems without buffering it processes tasks at the maximal rate of the stream (i.e. \( TP = 1 \)) with nearly the minimal number of processors ideally required, and it does so at the expense of the low turnaround time found in the other systems.
4-3 The Hybrid System

In the previous section, it was shown that a system of N processors can achieve nearly N fold improvement in throughput over a single processor system through the incorporation of N buffers into its interconnection network. It suffers significantly greater turnaround time than it would without these buffers, however. The hybrid system described here offers very nearly the throughput of the loop system and lower turnaround time than the systems without buffers by using between one and N buffers in its interconnection network and combining aspects of all of the previous systems.

The N processors are divided up among R buffers (1 ≤ R ≤ N) connected together to form a loop and interfaced to the input and output streams in the same manner as the loop system. Associated with each buffer i (1 ≤ i ≤ R) is an interconnection network without buffers, as described in chapter three, which connects the buffer to its processors. The buffer serves as the network's input register, and buffer i + 1 (or the output register when i = R) serves as the network's output register. In figure 4-10, a hybrid system with N = 4 and R = 2 is illustrated. This system has three processors connected to each register with a round robin interconnection network. In general, the processors can be subdivided in any manner among the registers. Notice that the hybrid system is modular, where a module consists of a subset of the processors, a round robin interconnection network, and a loop buffer.
The hybrid system's loop of buffers operates identically to the simple loop system; each processor subset and its respective interconnection network simply behaves like a fast processor to its loop buffer. Input packets enter the system at buffer 1, and output packets exit at buffer R in the usual manner. (See section 4-2 for a description of the loop system's operation).

Each buffer's interconnection network operates very similarly, but not identically (unless \( r = 1 \)), to the description in chapter three. The differences are as follows. First, input and output packets can only be transferred to and from the processors at integral multiples of \( r \) clock periods due to the synchronization imposed by the loop. Second, there may not always be an input packet in a network's input register, and its output register may not always be available; these events occur concurrently whenever the network's loop buffer contains an output packet.

In the following performance analysis of the hybrid system, we consider only the unordered round-robin (URR) control mechanism for each buffer's interconnection network. The ordered round robin (ORR) scheme preserves the order of the packet stream at the expense of performance with respect to the URR scheme; the loop buffers, however, negate this advantage by disrupting the packet order. The left priority first (LPF) scheme is clearly inferior to the URR scheme, and there is no analysis in chapter three to build on here.
Under the assumption that \( r = 1 \), the hybrid system is easily analyzed by directly combining the URR and loop analyses. Let \( N_i \) equal the number of processors allocated to buffer \( i \), so that \( \sum_{i=1}^{R} N_i = N \). Now, each processor in buffer \( i \)'s module gains access to the buffer exactly once every \( N_i \) clock periods and finds an input packet in the buffer with probability \( \ell_i \), where \( \ell_i \) is the probability that an input packet enters buffer \( i \) at any clock period, as previously defined in the recursive loop analysis. The behavior of each processor allocated to buffer \( i \) is modeled by a Markov chain with one \textit{ready} state and \( k \equiv \frac{h}{N_i} \) \textit{busy} states (see figure 3-7) as described in the analysis of the URR system. The transition probability, \( t_0 \), from the \textit{ready} to the \textit{busy} 1 state is modified from equation 3-8 as follows:

\[
t_0 = \ell_i \cdot [1 - P(N_i-1)] ;
\]

that is, the processor exits the \textit{ready} state if it finds an input packet in its buffer and if this packet's task is at least \( N_i \) clock periods long. It reenters the \textit{ready} state if it does not find an input packet or if the input packet corresponds to a task less than \( N_i \) clock periods long; this occurs with probability:

\[
1 - \ell_i + \ell_i \cdot P(N_i-1) = 1 - t_0 .
\]

Note that whenever a processor removes an input packet from its buffer, it places an output packet in the succeeding buffer,
except for processors allocated to buffer N, which place their output packets directly into the output register. The other transition probabilities are modified from equation (3-9); the probability, $t_j$, from the busy $j$ state to the busy $j+1$ state is:

$$
t_j = \frac{1 - P[(j+1) N_i - 1]}{1 - P[j N_i - 1]} \quad \text{for} \quad j = 1, 2, ..., k-1,
$$

(4-17)

where $N_i$ is substituted for $N$.

The processor moves from the busy $k$ state back to the ready state with probability $t_k = 1$, as before. Using equations (3-10), (4-16) and (4-17), we recalculate the state probability,

$$
\begin{align*}
\ell(i) &= \frac{1}{1 + \sum_{j=0}^{k-1} \{1 - P[(j+1) N_i - 1]\}} \\
&= \frac{1}{1 + \sum_{j=N_i}^{h} \sum_{j=N_i} \{Q[j N_i] P'(j)\} \quad .}
\end{align*}
$$

(4-18)

Since all processors associated with buffer $i$ find input packets in the buffer with the same probability, they have the same probability, $\ell(i)$, of residing in the ready state. Thus, the probability that the processor currently with access to buffer $i$ through the URR interconnection network is busy and will not draw an input packet out of buffer $i$ is:

$$
1 - \ell(i) \equiv \beta(i) \quad .
$$

(4-19)
Substituting this value for $\beta(i)$ into equation (4-3) and thereby using the same approximations needed in the loop analysis to derive this equation, we calculate the probability, $\ell_{i+1}$, that an input packet emerges from buffer $i$. The throughput of the hybrid system is then:

$$TP = 1 - \ell_{R+1} \quad (4-20)$$

since there are $R$ buffers on the loop.

The turnaround time of the hybrid system is easily calculated using the components defined for the loop system. The average time, $TT_\ell$, that the tasks' packets spend on the loop, including the time required for input to a processor and output from the loop to the output register (for processors 1, 2, ..., $R$-1) is just:

$$TT_\ell = \frac{R}{TP} \quad ,$$

using the same arguments as before. The average time, $TT_e$, that the tasks spend in execution is $p^*$ clock periods, and the average time, $TT_w$, that the tasks wait for output from the processors is given by equation (2-8):

$$TT_w = \frac{N}{TP} - (p^* + r) \quad .$$

The average output time, $TT_0$, either to the loop (for processors 1, 2, ..., $R$-1) or to the output register (for processor $R$) is trivially $r$ clock periods. Together, we have:
\[ \text{TT} = \text{TT}_L + \text{TT}_e + \text{TT}_w + \text{TT}_0 \]

\[ = \frac{R}{TP} + p^* + \frac{N}{TP} - (p^* + r) + r \]

\[ = \frac{N + R}{TP} \quad ; \] \hspace{1cm} (4-21)

This result is independent of the assumption that \( r = 1 \).

When \( R = 1 \) and \( r = 1 \), the hybrid system is identical to the URR system with \( N_1 = N \) processors. Using the above analyses, the throughput of this system is:

\[ TP = 1 - \ell_2 \]

\[ = \ell(1) = \frac{1}{h \sum_{j=N}^{j=N} Q[j] P'(j)} \]

which is identical to equation (3-11) for the throughput of a URR system with \( r = 1 \). Also, we have:

\[ \text{TT} = \frac{N + 1}{TP} \]

which conforms to equation (3-4). When \( R = N \) and \( r = 1 \), the hybrid system is identical to the loop system with \( N \) processors. We have \( N_1 = 1 \), and equations (4-18) and (4-19) yield:

\[ \beta(i) = \frac{\ell_i p^*}{1 + \ell_i p^*} \]

which conforms to equation (4-4) in the analysis of the loop system. Equation (4-20) becomes identical to equation (4-1), and thus this analysis correctly gives the throughput of the
loop system with \( r = 1 \). The turnaround time agrees also, since from equation (4-21) we have:

\[
TP = \frac{2N}{TP}
\]

which agrees with equation (4-7). Finally, when \( N = R = 1 \), the hybrid system is just a single processor system, and the performance analyses correspond here, as well.

The above recursive model for the hybrid system with \( r = 1 \) conforms closely to simulation of this system. The conformity improves as \( R \) approaches unity since fewer approximations from the recursive loop model are required, and the URR system model is exact. This is illustrated in figure 4-11 in which the performance measures are plotted versus the number of buffers in a hybrid system with \( N = 60 \), \( p^* = 55.5 \), and \( r = 1 \), using values obtained from both the recursive model and simulation with uniform and geometric distributions of task execution times. Note that although continuous curves are drawn in this figure, only values of \( R \) that divide \( N \) evenly correspond to actual systems in which each buffer shares an equal number of processors.

In figures 4-12 and 4-13 respectively, simulation results for the throughput and turnaround time are plotted versus the number of processors in the same hybrid system with \( r = 1 \), and \( p^* = 55.5 \) and a uniform distribution of task execution times. The figures show a family of curves, corresponding to
FIGURE 4-11: Throughput and Turnaround Time Versus Number of Buffers for the Hybrid System
\[ r=1, \; N=60, \; p^*=55.5, \; \text{uniform distribution} \]
FIGURE 4-12: Throughput Versus Number of Processors for the Hybrid System
r=1, p*=55.5, uniform distribution

Throughput, TP

Number of processors, N
FIGURE 4-13: Turnaround Time Versus Number of Processors for the Hybrid System
r=1, p*=55.5, uniform distribution

- ○...R=1 (URR System)
- ○...R=10
- □...R=2
- ▽...R=N/2
- △...R=5
- ★...R=N (Loop System)
several values for $R$, namely 1, 2, 5, 10 and $N/2$; the processors are divided equally among the buffers. The curves with $R = 1$ correspond to the URR system, and the curves with $R = N$ correspond to the loop system. A significant improvement in throughput is obtained with a small increase in $R$, especially when $N$ is close to $p^*$. When $R$ equals $N/2$, the throughput is always within five percent of the values for $R$ equal to $N$. The turnaround time behaves in a unexpected but desirable manner as $R$ increases. When $R$ equals 2, 5, and 10, the curves dip below the curve with $R$ equal to 1. That is, the addition of a few buffers to the system actually improves the turnaround time instead of degrading it, as expected from the behavior of the loop system ($R=N$). This effect is most pronounced when $N$ is close to or slightly larger than $p^*$. When $R$ equals $N/2$, the turnaround time behaves like that for the loop system, remaining much higher than the $R = 1$ curve for all values of $N$; it also has the characteristic bend upwards at $N \approx p^*$.

In order to clarify the above observations, refer again to figure 4-11, in which the performance measures are plotted versus the number of buffers with $N = 60$, the value for $N$ at which the effects of varying $R$ are most pronounced for $p^* = 55.5$. The throughput rises rapidly from the URR system's value as buffers are added, reaching 0.95 when $R$ equals 20. The turnaround time drops rapidly, reaching a minimum when $R$ equals 10 and rising above the URR system's value only after.
R equals 30. The turnaround time improves with the addition of a few buffers because the added throughput due to these buffers offsets the added delays that they create (recall equation (4-21)). Only after the incremental improvement in throughput with added buffering diminishes does the buffering's delay dominate the turnaround time.

The above results indicate that when \( N \approx p^* \), the hybrid system offers a substantial improvement in both throughput and turnaround time over the URR system. Also, this system offers nearly throughput of the loop system with many fewer buffers and thus much lower turnaround time. The URR system alone provides the lowest turnaround time of these three systems when \( N \) is very small, and the loop system always provides the highest throughput for a given \( N \). However, the hybrid system clearly provides the best combination of both performance measures for \( N \approx p^* \) and appropriate \( R \). The optimal value of \( R \) for a given system is difficult to determine in general due to the complexity of the analysis.
CHAPTER V
APPLICATIONS, EXTENSIONS, AND CONCLUSIONS

5-1 Applications

In this section we suggest the use of the packet interconnection networks developed in chapters three and four as a scheduling mechanism for a dedicated database management system. We first briefly review the functions and attributes of a database management system and outline the motivations for its implementation as a special purpose architecture. We then specify how this research might contribute to its design.

A database management system (DBMS) can be defined as a "software system whose main objective is to provide a convenient way to access and modify large amounts of stored data. It provides an environment in which a user can represent and view information of the real world with complex logical relations." [GHOS 76] According to Baum and Hsiao [BAUM 76], a database management system should:

(a) provide large on-line storage capacity (typically between $10^8$ and $10^{10}$ bytes),

(b) support a high level data model (e.g. the relational, hierarchical, and network models described by Date [DATE 75]) and an appropriate query and access control language,

(c) support shared, concurrent user access with adequate response time and with security and integrity enforcement,

(d) be high reliable.
The users of a DBMS normally perform the following operations (perhaps among others) [WIED 77]:

(a) creation or destruction of databases,

(b) searching for selected items (often specified by predicates),

(c) updating of selected items,

(d) insertion of new items,

(e) removal of selected items,

(f) reorganization of the database.

Most database management systems are incorporated into the architecture of a general purpose computer system. Canaday, Harrison, et al have recognized several advantages in the use of a special purpose architecture, or "backend machine," to support a DBMS [CANA 74]. This approach offers certain economy through its specialization, affords the sharing of the DBMS by multiple systems, and facilitates its protection from failure and unauthorized use. When implemented with a single powerful processor, it has the disadvantages of higher hardware cost (although perhaps lower software costs), the loss of flexibility in the balancing of the total hardware resources, and added delays in the transmission of data between the backend machine and the main system.

The use of multiple microprocessors potentially alleviates these problems and makes possible the design of database machine with high performance and reasonable cost. Slotnick [SLOT 70] proposed the incorporation of logic (eg. microprocessors) into
each head of a head per track rotating storage (see figure 5-1). This architecture enables the realization of very large, although slow, associative memories and has been used in several data base machines including RAPID [PARH 72], CASSM [SU 75], RAP [OZKA 74], and RARES [LIN 76]. Lin, Smith, et al [LIN 76] review differences among these designs. The central processor to which these machines are connected by a data channel is usually responsible for the:

(a) support of data communication with the users,
(b) completion of user queries into machine commands,
(c) transfer of the machine commands and data to the backend machine,
(d) control of data base security and integrity. [OZKA 74]

Commands are broadcast to all processors which search for specific items in the rotating storage. The processors output their results over a common bus back to the central processor. The time to search an entire database is reduced to the time required to search one track of the rotating storage and thus is independent of the size of the data base. A controller may be used between the central processor and the logic elements depending on the amount of processing performed at the central processor and the complexity of the logic elements. Baum and Hsiao [BAUM 76] and others propose that electronic memories such as charge coupled devices (CCD's) and magnetic bubble memories be used to implement the rotating storage since they offer one or two orders of magnitude improvement in access time over fixed head disks. They do not feel that this architecture is practical for databases larger than.
FIGURE 5-1: Block Diagram for "Logic-per-track" Database Architecture
$10^8$ bytes, presumably due to the large number of processors and the effects of their mutual contention for the data channel. They suggest that the architecture be used for the processing of directory information only and be coupled to the actual database held on moving head disks containing a small number of processors; this concept is developed in the design of the database computer (DBC) at Ohio State University [HSIA 76].

The interconnection schemes developed in this thesis can potentially enhance the capabilities of the above database architecture by allowing concurrent access from multiple users. Certain applications, such as an airline's reservations system or a department store's inventory, require the concurrent processing of a large number of simple, independent "retrieval" and "update" transactions from many users; the amount of data transferred in each of these transactions is relatively small. Hence, the commands generated by the central processor and the data output by the database machine processors can be encoded in packets of reasonable size. Multiple sets of "logic per track" (or multiple tracks) processors described above can be placed at regular intervals around the tracks, as shown in figure 5-2, and interconnected to the data channel with a hybrid system (described in chapter four). Commands can then be routed to the "processor sets"

*A database of this size would contain 6,250 tracks at 16K bytes per track. Associating a processor with 64 tracks (8 x 16K bytes) requires about 98 processors.*
FIGURE 5-2: Block Diagram for "Logic-per-track" 
Database Architecture with Multiple Processor Sets
in the form of input packets, and the results of the transactions are returned to the channel as output packets. Each command is broadcast to all of the processors in a given set by the controller and executed in the usual manner. Note that the execution time for the retrieval or updating of a single item is highly variable due to the rotation of the database; the rotational delay required to reach a specific item obeys a uniform distribution with the average equal to one half the total rotation time. The controller generates an output packet upon completion of the transaction and delivers it to the hybrid system for return to the data channel. The use of the hybrid system assures that the data channel runs at its maximal rate with fewer processor sets and lower average turnaround time per task than obtainable with a single two way bus. (The number of buffers used in the interconnection network depends of course on the parameters of the system.)

The processors have independent access to the entire database and are mutually excluded from concurrently changing the same location due to their physical placement. If more than one rotation of the data base is required to complete a transaction, then a processor set can set a flag in the database to prevent another processor set from altering the location during the transaction. Since the commands originate from different users of the database, they are independent and can be concurrently processed. The central processor is responsible for maintaining precedence relationships of multiple commands from a single user in the issuing of input packets. It is also responsible for
input/output with perhaps many users. Note that the output packets arrive at the central processor in random order. This may present significant software overhead and/or require special purpose hardware.

To explore the feasibility of this architecture, consider the following example. Assume that a database of $10^8$ bytes is divided into 6,250 16K byte tracks implemented on CCD's. Assume that 64 tracks are associated with one processor so that a processor set contains about 98 processors. The tracks are shifted at 10 Mhz and thus require 12.8 msec per revolution. If a simple "retrieval" or "update" transaction is completed within one rotation, then its average execution time is approximately 6.5 msec. (Allowing 1 msec for completion of the transaction once the desired location reaches the read heads). Assume that the data channel delivers four bytes in parallel at 1 M band and that packets are 64 bytes long; then a packet is transferred every 16 μsec.

Hence at least 406 processor sets are required in order to saturate the channel (i.e., achieve maximal throughput; see chapter two), corresponding to 316 bits per track for each processor set.

By associating a 64 x 316 = 20,224 bit CCD memory with each processor, this architecture could be constructed with 406 x 98 = 39788 (!) processor-memory modules arranged in a two dimensional array.

The interconnection schemes developed in this thesis can also be used within each processor set in order to reduce contention for the bus located between the "logic per track" processors and the controller. (See figure 5-3.) This contention arises when a
FIGURE 5-3: Detailed Block Diagram of a Processor Set
single command generates several matches in the database and, hence, many output packets. This would occur in applications (unlike those above) such as searches into a telephone directory. Note that commands are broadcast to the processors, and hence no input packets are used here. This situation is not modeled in this thesis, but one would expect that the buffering techniques described in chapter four to improve the throughput of the output packets and reduce the average turnaround time of the transactions. A $10^8$ byte data base with the more conventional single processor set could be constructed with 98 processor-memory modules containing approximately $10^6$ bytes of CCD memory in each module, arranged in 64 16K byte tracks.
5-2 Extensions

The single stream processing systems presented here can be generalized in a relatively simple manner by the use of multiple packet streams to connect the multiprocessing system to the central processor; this increases the throughput of the system beyond that obtainable from a single stream. The multiple streams are readily incorporated into the URR system by circulating multiple control bits in the control loop, one for each stream; in the loop and hybrid systems, the streams are simply interfaced to different buffers. These schemes are illustrated in figure 5-4. In this section we investigate the performance of these architectures and compare it to the performance of the single stream systems analyzed earlier.

Assume that there are \( N \) processors and \( S \) streams such that \( S \) divides \( N \); let \( N_S = N/S \). Further assume that the central processor imposes no bottlenecks or any stream; i.e. the streams operate at the maximal possible rate (see section 2-2). Also assume that output packets can return to the central processor by any stream, not necessarily the stream in which their corresponding input packets arrived. We redefine the performance measures as follows:

\[
TP \equiv \sum_{j=1}^{S} TP(i) , \text{ and} \\
TT \equiv \frac{1}{S} \sum_{j=1}^{S} TT(i) , \text{ for } j = 1,2,\ldots,S
\]
FIGURE 5-4a: Block Diagram of URR System with Multiple Streams
FIGURE 5-4b: Block Diagram of Loop System with Multiple Streams
where \( TP(j) \) and \( TT(j) \) are the performance measures for stream \( j \); \( TT(j) \) is measured for the output packets returning in stream \( j \) to the central processor. The throughput is the total throughput obtained from the streams, and the turnaround time is the average turnaround time for all tasks in the system.

The performance of the URR system with \( S \) streams and \( r = 1 \) is computed as follows. The control bits are placed at regular intervals around the control loop so that each processor has access to one of the streams every \( N_s \) clock periods, and can receive a task at these times. Hence, the state probability that each processor is in the \textit{ready} state is given by equation (3-11):

\[
1 = \frac{h}{1 + \sum_{i+N_s} Q[i] P'(i) N_s}
\]

where \( N_s \) is substituted for \( N \). This state probability is clearly the throughput of each stream, and thus the aggregate throughput of the system is

\[
TP = S \cdot 1
\]

The turnaround time is computed in the same manner as before.

The average time, \( TT_i(j) \), that a task returning by stream \( j \) spends as input packet is:

\[
TT_i(j) = t_f + r = \frac{1}{TP(j)} = \frac{S}{TP}, \text{ for } j = 1,2,...,S
\]
the average time, $\mathbf{TT}_e$, spent in execution is just $p^*$ clock periods; the average time, $\mathbf{TT}_w$, spent waiting for output is given by equation (2-8) where $\mathbf{TP}$ is the aggregate throughput of the system:

$$\mathbf{TT}_w(j) = \frac{N}{\mathbf{TP}} - (p^*+r), \text{ for } j = 1, 2, \ldots, S;$$

finally the average time, $\mathbf{TT}_o$, that a task spends as an output packet is $r$ clock periods. Note that all components of $\mathbf{TT}$ are independent of the stream by which the output packets return to the central processor. Hence, the total turnaround time is just:

$$\mathbf{TT} = \frac{S}{\mathbf{TP}} + p^* + \frac{N}{\mathbf{TP}} - (p^*+r) + r$$

$$= \frac{N+S}{\mathbf{TP}}$$

$$= \frac{N_S+1}{\mathbf{TP}(j)} \text{ for } j = 1, 2, \ldots, S.$$

Notice that the performance of this system is the same as the combined performance of $S$ single stream systems with $N_S$ processors per system. The latter system has the advantage that both input and output packets for a given task use the same stream. If this requirement is imposed on the $S$-stream system, then its performance is degraded because processors have to wait for a particular control bit in order to output results, instead of waiting for any control bit. Furthermore, additional hardware is required to distinguish the control bits.
The performance of the loop and hybrid systems with $S$ streams and arbitrary packet transfer times is also easily calculated. We assume that there are $N$ processors and $R$ buffers in the system with $N_R = N/R$ processors per buffer and $R_S = R/S$ buffers per stream (i.e. $S$ divides $R$, and $R$ divides $N$). Stream $j$ is connected to the loop so that input packets enter the loop at buffer $1 + (j-1)R_S$, (the stream's input register) and output packets exit the loop (and enter the stream's output register) after reaching buffer $jR_S$, for $j = 1,2,...,S$.

Note that module* $jR_S$ deposits its output packets directly into the stream's output register. Define parameter $\ell_{i+1}^i$ as the probability that an input packet exits buffer $i$, for $i = 1,2,...,R$. As before, $\ell_i$ is the probability that an input packet enters buffer $i$ and hence is the probability that buffer $i$ holds an input packet. It is the case that $\ell_i^1 = \ell_i$ for all buffers that are not input registers, i.e. for buffer $i$ where $i = 1,2,...,R$ and $i \neq 1 + (j-1)R_S$ for $j = 1,2,...,S$, since the former buffers only receive input packets from the preceding buffers on the loop; the input registers receive input packets both from the preceding buffers and the stream. Because the streams deliver input packets to the loop at the maximal possible rate, then the input registers contain input packets at all times, and we have:

$$\ell_1 + (j-1)R_S = 1 \quad \text{for} \quad j = 1,2,...,S .$$

*The term module can be interpreted as the URR system of $N_R$ processors connected to each loop buffer.
Using equation (4-3) and (4-18), with $N_i = N_R$ for $i = 1, 2, \ldots, R$ we can compute the state probability that the modules connected to these buffers are idle, and the probability that input packets exit these buffers:

$$\ell'_{2+(j-1)R_S} = \ell_{1+(j-1)R_S} \cdot \beta(1+(j-1)R_S)$$

$$= \beta(1)$$

$$= \ell_{2+(j-1)R_S} \quad \text{for} \quad j = 1, 2, \ldots, R_S$$

This can be repeated for successive buffers on the loop:

$$\ell'_{i+1} + (j-1)R_S = \ell_{i+(j-1)R_S} \cdot \beta(i+(j-1)R_S)$$

$$= \ell_{i+1} \cdot \beta(i)$$

$$= \ell_{i+1+(j-1)R_S} \quad \text{for} \quad i = 1, 2, \ldots, R_S-1$$

and $j = 1, 2, \ldots, S$;

in each case we find that:

$$\ell'_{i} = \ell'_{i+(j-1)R_S} \quad \text{for} \quad i = 1, 2, \ldots, R_S-1, \quad \text{and} \quad j = 1, 2, \ldots, S$$

Finally, we obtain the probability $\ell'_{j \cdot R_S}$ that input packets exit buffer $jR_S$ and enter buffer $1 + (jR_S \mod R)$ thus prohibiting stream $j$ from placing a new input packet on the loop,

$$\star \equiv x \mod y = x - \left\lfloor \frac{x}{y} \right\rfloor y$$
for \( j = 1,2,\ldots,S \):

\[
\ell'_{jR_S} = \ell'_{jR_S-1} \cdot \beta(jR_S - 1)
\]

\[
\ell'_{R_S} \cdot \beta(R_S) \quad \text{for} \quad j = 1,2,\ldots,S
\]

The throughput for stream \( j \) then is:

\[
TP = \frac{1 - \ell'_{jR_S}}{r} = \frac{1 - \ell'_{R_S}}{r} \quad \text{for} \quad j = 1,2,\ldots,S
\]

All streams have the throughput of a single stream hybrid system with \( R_S \) processors. The aggregate throughput of the system is simply:

\[
TP = S \cdot \frac{1 - \ell'_{R_S}}{r}
\]

The calculation of the turnaround time follows the same reasoning as the earlier derivation for the single stream loop and hybrid systems in section 4-2. It is subdivided into four components:

\( TT_L \), the average time that a packet spends on the loop as either an input or output packet, including the time required for input to a processor and output from the loop to the output registers (this last sub-component does not apply to module \( jR_S \), for \( j = 1,2,\ldots S \) which deposits output packets directly into the output register).

\( TT_e, TT_w \), as defined in section 4-2, and
the average time required to output an output packet from a processor either to the loop (for modules other than module $jR_\Sigma$, for $j = 1, 2, \ldots, S$) or directly to the output register (for modules $jR_\Sigma$, for $j = 1, 2, \ldots, S$).

The first component is a multiple of $R_\Sigma r$ clock periods. Consider module $i + (j-1) R_\Sigma$ for $i = 1, 2, \ldots, R_\Sigma - 1$ and $j = 1, 2, \ldots, S$. Input packets reaching this module require $(i-1)r$ clock periods to reach the module's buffers from the nearest buffer $1 + (j-1)R_\Sigma$, for $j = 1, 2, \ldots, S$ (i.e. input register). Now, the input packet, upon entering the loop, may circulate through $k$ input registers, for positive integer $k$, before a module takes it, and this requires $kR_\Sigma r$ clock periods. The input packet enters the module in $r$ clock periods, and after the task is executed, the output packet appears in buffer $i + 1 + (j-1)R_\Sigma$. Then $(R_\Sigma - (i+1))r$ clock periods are required for the packet to reach buffer $jR_\Sigma$, whence it exits the loop and enters stream $j$'s output register in $r$ clock periods. In total, we have:

$$TT_{\Sigma} = (i-1)r + kR_\Sigma r + [R_\Sigma - (i+1)]r + r$$

$$= (k+1)R_\Sigma r \quad (5-1)$$

for this module. Input packets reaching module $jR_\Sigma$ for $j = 1, 2, \ldots, S$ require $(R_\Sigma - 1)r$ clock periods to reach its buffers after circulating in the loop for $kR_\Sigma r$ clock periods in the
loop (as above) and require \( r \) clock periods to enter the module. For this module, we have:

\[
TT_L = (R_S - 1)r + kR_Sr + r
\]

\[
= (k+1) R_Sr
\]

since output packets from this module do not enter the output register from the loop. Now, if the system is active for \( T \) clock periods, then in the limit as \( T \) grows without bound, \( S \cdot \ell_R \) (\( T/r \)) input packets enter input registers from the preceding buffers, and \( TP \cdot T \) tasks are completed; hence, the average number of times that input packet circulate past a stream buffer is:

\[
\lim_{T \to \infty} \frac{1}{r} S \cdot \ell_R \frac{T}{TP} + \frac{S \ell_R}{r TP} = k.
\]

Substituting this value for \( k \) into equation (5-1), we have:

\[
TT_L = \frac{S \ell_R}{r TP} + 1 \] \( R_S \cdot r = \frac{S}{1 - \ell_R} \) \( R_Sr = \frac{S R_S}{TP} \).

The other components of the turnaround are as follows:

\[
TT_e = p^*
\]

\[
TT_w = \frac{N}{TP} - (p^* + r),
\]

\[
TT_o = r.
\]
Since all components are independent of the stream used by tasks, the total turnaround time is just:

\[
TT = TT_L + TT_e + TT_w + TT_o
\]

\[
= \frac{SR_S}{TP} + p^* + \frac{N}{TP} - (p^*+r) + r
\]

\[
= \frac{N+SR_S}{TP}
\]

\[
= \frac{N_S + R_S}{TP(i)}
\]

The turnaround time is identical to that for a single stream hybrid system with \(R_S\) registers and \(N_S\) processors. Hence, the performance of a hybrid system with \(S\) streams is the same as the combined performance of \(S\) single stream systems with \(N_S\) processors and \(R_S\) registers per system. The use of \(S\) single stream systems has the advantage that packets enter and exit the system in the same stream. If this constraint is imposed on the \(S\) stream system, then the performance degrades since output packets have to circulate to specific buffers in order to exit the system, instead of simply circulating to any buffer \(jR_S\), for \(j = 1, 2, ..., S\). Additional hardware to implement this feature is required, also.

The above analysis shows that multiple packet streams can readily be incorporated into the architectures presented in this thesis. When the streams operate at the maximal possible rate, there is no performance advantage in so doing
over the use of separate single stream systems with the same aggregate number of processors. It is reasonable to expect that this is not the case in general since the former approach allows every stream to access every processor, which may be advantageous when the streams are delivering tasks at varying rates.

* * *

A disadvantage of the loop system is that the tasks experience highly variable delays in waiting for output from the processors. Simulation shows that tasks executed by processors close to processor \( N \) encounter much longer delays in this regard that do processors close to processor 1, due to the higher probability that output packets enter their buffers. A simple modification of the loop system alleviates this problem without significantly affecting the performance. All buffers are allowed to serve as input registers, and the input and output streams make use of alternate buffers at different transfer intervals, as determined by a control loop with the control bit circulating opposite to the direction of packet flow. This architecture is illustrated in figure 5-5, and its performance when \( p^* = 55.5 \) and \( r = 1 \) is shown in figure 5-6. Simulation shows that all processors experience similar delays in waiting for output; the distribution of the processing load is evenly balanced also (i.e. the \( \beta \)'s are nearly equal).
FIGURE 5-6: Throughput and Turnaround Time Versus Number of Processors for the Enhanced Loop System
r=1, p*=55.5, uniform and geometric task execution times
* * *

In the previous chapters, the performance analyses and comparisons were restricted to systems with packet transfer times equal to unity (i.e. \( r=1 \)). This is the upper bound on the throughput allowed by the stream and thus on the performance obtainable from the systems. When this restriction is generalized (i.e. for \( r=2,3,\ldots \) clock periods), the advantages obtained through the use of buffering diminish, and the performance of the loop and round-robin systems converges. The probability that the round-robin control mechanism can locate a free processor during a transfer interval increases with increasing \( r \) since multiple (i.e. \( r \)) processors can be "scanned" in one transfer interval, and hence the performance improves relative to systems with buffering; the absolute performance is degraded of course, since \( TP=1/r \) and \( TT \approx 1/TP \). For example, figures 5-6 and 5-7 show the performance of the ORR, URR, and loop systems with \( r=2 \) and \( p^*=55.5 \) (task execution times uniformly distributed). Note that here the URR system is actually better than the loop system for \( N \) less than 50 processors.
FIGURE 5-7: Throughput Versus Number of Processors for the ORR, URR, and Loop Systems

r=2, p*=55.5, uniform distribution of task execution times

- ■ = ORR system
- ● = URR system
- ▲ = Loop system
FIGURE 5-8: Turnaround Time Versus Number of Processors for the ORR, URR, and Loop Systems
$r=2$, $p^*=55.5$, and uniform distribution of task execution times

- □ = ORR system
- ○ = URR system
- ▲ = Loop system
5-3 Suggestions for Further Research

In this section we suggest several aspects of this work open to further research.

First, there is a pressing need for further refinement of the modeling techniques employed here. The ORR system's model and the URR system's general model require too many states to be practical. There is reason to believe that redundant information might be eliminated by exploiting the symmetry inherent in these systems. This symmetry occurs because the control mechanism favors no particular processor in its distribution of tasks (unlike the control mechanism for the LPF system). Other evidence of redundancy in the URR system's general model lies in the fact that the number of states is unchanged if \( r = 1 \), and yet the simple model demonstrates that the throughput can be expressed by a simple summation in this case (equation (3-11)). The two approximations made in the recursive analysis of the loop and hybrid systems should be explored; they are related since they both result from the same statistical dependency, and perhaps they could be improved if they are more fully understood.

Several useful generalizations of these models deserve consideration. The modeling of the central processor should be generalized in order to lift the assumptions that it delivers input packets at every opportunity and receives output packets immediately after they arrive in the output register. Perhaps the central processor could be modeled with a Markov chain
integrated into the previous models so that one could specify
the probability distributions for the input packets'
interarrival times and the output register's service times.
It would be desirable to allow variable packet lengths, also
specified by a probability distribution, as might occur in
realistic situations. The buffers would have to be the length
of the longest allowable packet, however, and an asynchronous
communication scheme would be required if the packets were
transmitted serially (and hence the transmission delays were
not uniform for all packets). The models should also be
extended to predict second order statistics (i.e. variances)
for the performance measures. There may be applications for
these systems in which the average turnaround time, for example,
may be acceptable and yet some tasks experience intolerably
high turnaround times; knowledge of second order statistics
would be vital for these applications. These generalizations
introduce more complexity into modeling techniques that are
already intractable. The relative merits of modeling and
simulation for the analysis of multiprocessing systems should
be studied.

Finally, further research is needed to extend the
architectural concepts introduced and studied here. Other
buffering strategies such as a tree structured array of buffers
(see figure 5-9) should be considered and their performance
compared to the schemes presented here, especially for the
connection of multiple streams to the processors. The software
FIGURE 5-10: Block Diagram of a System with Tree-structured Buffers
requirements for the formulation of the input packets and collection of output packets at the central processor have not received attention here, and certainly will present significant problems.
5-4 Conclusions

In this thesis, we have presented several methods (implemented in hardware) for the scheduling of a large number of processors. These processors are organized into a special purpose multiprocessing system connected by two data channels to a general purpose computer system (called the central processor). Fully independent tasks are formulated and delivered to the system over one channel as a single stream of input packets. The packets are routed to the processors by an interconnection network containing zero or more buffers. The processors execute the tasks and output the results as output packets which are routed to the second channel and then to the central processor in another stream.

Two performance measures, the throughput (TP) and the average turnaround time (TT), were defined and used to compare the merits of the systems presented. Probabilistic models were developed for these systems under the assumption that the task execution times vary according to a known but arbitrary probability distribution and that the central processor introduces no bottlenecks (see section 2-3). The performance of the systems was measured by these modeling techniques and by simulation, using both geometric and uniform distributions for the task execution times.

The systems presented in this thesis are divided into two categories: those with interconnection networks containing one or more buffers. In the first category, the interconnection networks consist simply of two busses connecting the processors to the two
data channels. Three control mechanisms for the concurrent arbitration of both busses are compared, two variations of the simple "round-robin" scheme (called the ordered round-robin (ORR) and the unordered round-robin (URR systems), and the left priority first (LPF) scheme found in many computer systems; the ORR system preserves the order of the packet stream while the URR system does not. In the second category, the interconnection networks consist of a circular arrangement of buffers through which the packets continuously circulate. In the loop system, there are as many buffers as processors, and in the hybrid system multiple processors are associated with each buffer and connected to it with a URR system.

The performance analysis shows that the loop system achieves very nearly the ideal N-fold improvement in throughput over that for single processor systems, due to its ability to concurrently schedule multiple processors. The round-robin systems achieve this throughput with many more processors when the task execution times are highly variable (the URR system being slightly better than the ORR system in this respect), and the LPF system never achieves this throughput. However, the loop system suffers considerably higher turnaround time than these other systems for a given number of processors, due to the delays imposed on the packets by its buffers. The use of an intermediate amount of buffering in the hybrid system results in a minimal loss in throughput from the loop system and offers a lower turnaround time than the round-robin systems. It provides an effective mechanism for the scheduling of a large number of processors.
These architectures are extended to accommodate multiple packet streams, so that the throughput can be increased beyond the limitations of a single stream and the resulting performance is calculated.

Finally, the applicability of these systems to the design of dedicated database management systems is studied. These architectures can potentially enhance "logic per track" architectures and/or provides concurrent access to the database by many users.
APPENDIX

Theorem 4-2:

\[ k_{i,j} \leq (i-2)^j \]

Proof:

Proof is by double induction on \( i \) and \( j \).

(Basis step: \( i = 2 \) and \( j = 0 \)) We have \( k_{2,0} = 1 \leq (i-2)^j = 0^0 \).

(Induction step) Assume that \( k_{i,j} \leq (i-2)^j \), and show that \( k_{i+1,j+1} \leq (i-1)^{j+1} \). (Subinduction step I) Show that \( k_{i+1,j} \leq (i-1)^j \).

By substituting equation (4-14) into equation (4-13), we have:

\[ k_{i+1,j} \leq (i-2)^j + \sum_{\ell=0}^{j-1} (i-2)\ell (i-2)^{j-1-\ell} \]

\[ \leq (i-2)^j + j(i-2)^{j-1} \text{ for } j = 1,2,\ldots,2^{i-2}-1, \]

and

\[ k_{i+1,j} \leq \sum_{\ell=1}^{2^{i-2}-1} (i-2)\ell (i-2)^{2^{i-2}-1+\ell} \]

\[ \leq 2^{i-2} (i-2)^j \]

\[ \leq j(i-2)^{j-1} \text{ for } j = 2^{i-2}, 2^{i-2}+1,\ldots,2^{i-1}-1. \]

We will show that \( (i-2)^j + j(i-2)^{j-1} \leq (i-1)^j \) which proves this sub-induction step. Using the binomial theorem, we have:

\[ (i-1)^j = [(i-2) + 1]^j = (i-2)^j + j(i-2)^{j-1} + T_1 \]

\[ \geq (i-2)^j + j(i-2)^{j-1}, \]
where $T_1$ are positive terms of order $(i-2)^{j-2}$. Hence, it follows that $k_{i+1,j} \leq (i-1)^j$. (Sub-induction step II) Show that $k_{i,j+1} \leq (i-2)^{j+1}$. First, we eliminate the recursion in equation (4-13) as follows:

$$k_{i+1,j} = \sum_{m=L_2}^{i} \sum_{\ell=0}^{j-1} k_{m,\ell} k_{m,j-1-\ell} + \sum_{\ell=L_3}^{L_2} k_{L_2-1,\ell} k_{L_2-1,\ell} V_1 V_1 + L_3 - \ell$$

(A-1)

where $L_2 = 2 + \lceil \log_2 (j+1) \rceil$, $L_3 = \lceil \log_2 \frac{L_2-3}{2} \rceil$, and $V_1 = 2^{L_2-3} - 1$ since (1) for $m = L_2, L_2 + 1, \ldots, i$, we have:

$$k_{m+1,j} = k_{m,j} + \sum_{\ell=0}^{j-1} k_{m,\ell} k_{m,j-1-\ell}$$

according to equation (4-13) where $L_2$ in the smallest positive integer such that $j \leq 2^{L_2-2} - 1$, and (2) for $m = L_2 - 1$, we have:

$$k_{m+1,j} = \sum_{\ell=L_3}^{L_2-3} k_{m,\ell} k_{m,j-1-\ell}$$

$V_1 + L_3 - \ell$

according to equation (4-13) where $2^m = 2^{L_2-3} \leq j \leq 2^{L_2-2} - 1 = 2^{m-1} - 1$.

Second, we change the indices $i$ and $j$ in order to express equation (A-1) in terms of $k_{i,j+1}$, substituting $i$ for $i+1$, we have:

$$k_{i,j} = \sum_{m=L_2}^{i-1} \sum_{\ell=0}^{j-1} k_{m,\ell} k_{m,j-1-\ell} + \sum_{\ell=L_3}^{L_2} k_{L_2-1,\ell} k_{L_2-1,\ell} V_1 V_1 + L_3 - \ell$$

for $i = 3, 4, \ldots, N$ and $j = 1, 2, \ldots, 2^{i-2} - 1$;
substituting \( j + 1 \) for \( j \), we have:

\[
k_{i,j+1} = \sum_{m=L_4}^{i-1} \sum_{\ell=0}^{j} k_{m,\ell} \cdot k_{m,j-\ell} + \sum_{\ell=L_5}^{V_2} k_{L_4-1,\ell} \cdot k_{L_4-1,V_2+L_5-\ell}
\]

(A-2)

for \( i = 3,4,\ldots, N+L \) and \( j = 0,1,\ldots, 2^{i-2} - 2 \)

where

\[
L_4 = 2 + \left[ \log_2(j+2) \right], \quad L_5 = \left[ j+1 / 2 \right], \quad \text{and}
\]

\[
V_2 = 2^{L_4-3} - 1
\]

Now, it follows that:

\[
k_{i,j+1} \leq \sum_{m=L_4}^{i-1} \sum_{\ell=0}^{j} (m-2)^\ell (m-2)^{j-\ell} + \sum_{\ell=L_5}^{V_2} V_2+L_5-\ell
\]

\[
\leq \sum_{m=L_4}^{i-1} (j+1)(m-2)^{j} + (V_2-L_5+1) \cdot V_2+L_5
\]

\[
\leq \sum_{m=L_4}^{i-1} (j+1)(m-2)^{j} + (j+1)(L_4-3)^{j} = \sum_{m=L_4-1}^{i-1} (j+1)(m-2)^{j}
\]

(A-3)

by substituting equation (4-14) into equation (A-2) and since

\[
2^{-3} \leq j \leq 2^{-2} - 1,
\]

which implies that \( 2^{L_4-3} \leq j + 1 \leq 2^{L_4-2} - 1 \) and \( V_2 + L_5 = j \). By the binomial theorem, we have:

\[
(i-2)^{j+1} = ((i-3) + 1)^{j+1} = (i-3)^{j+1} + (j+1)(i-3)^{j} + T_1,
\]

and:

\[
(i-3)^{j+1} = ((i-4) + 1)^{j+1} = (i-4)^{j+1} + (j+1)(i-4)^{j} + T_2
\]
and so on, where $T_1$ and $T_2$ are positive terms. Thus, we have:

$$(i-2)^{j+1} \geq \sum_{m=2}^{i-1} (j+1)(m-2)^j,$$

and it follows from equation (A-3) that:

$$(i-2)^{j+1} \geq k_{i,j+1}$$

since $L_4 - 1 = 1 + \log_2(j+2) \geq 2$. This proves sub-induction step II. The induction step follows immediately from sub-induction steps I and II; which proves the theorem.

Q.E.D.
REFERENCES


