Telecommunication Networks and Computer Systems

Thomas G. Robertazzi

Computer Networks and Systems: Queueing Theory and Performance Evaluation

Springer-Verlag

TELECOMMUNICATION NETWORKS AND COMPUTER SYSTEMS

Series Editors

Mario Gerla
Aurel Lazar
Paul Kuehn

Thomas G. Robertazzi

Computer Networks and Systems: Queueing Theory and Performance Evaluation

With 92 Illustrations

Springer-Verlag
New York Berlin Heidelberg London
Paris Tokyo Hong Kong Barcelona

Thomas G. Robertazzi
Department of Electrical Engineering
State University of New York at Stony Brook
Stony Brook, NY 11794-2350
USA

Series Editors

Mario Gerla
Department of Computer Science
University of California
Los Angeles, CA 90024, USA

Aurel Lazar
Department of Electrical
Engineering
and Center for
Telecommunications Research
Columbia University
New York, NY 10027, USA

Paul Kuehn
Institute of Communications
Switching and Data Technics
University of Stuttgart
D-7000 Stuttgart
Federal Republic of Germany

Printed on acid-free paper.

© 1990 by Springer-Verlag New York Inc.
All rights reserved. This work may not be translated or copied in whole or in part without the written permission of the publisher (Springer-Verlag, 175 Fifth Avenue, New York, NY 10010, USA), except for brief excerpts in connection with reviews or scholarly analysis. Use in connection with any form of information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed is forbidden.
The use of general descriptive names, trade names, trade marks, etc. in this publication, even if the former are not especially identified, is not to be taken as a sign that such names, as understood by the Trade Marks and Merchandise Marks Act, may accordingly be used freely by anyone.

Camera-ready text prepared by the author using troff.
Printed and bound by: R.R. Donnelley and Sons, Harrisonburg, Virginia.
Softcover reprint of the hardcover 1st edition 1990

9 8 7 6 5 4 3 2 1

ISBN-13:978-1-4684-0387-9 e-ISBN-13:978-1-4684-0385-5
DOI: 10.1007/978-1-4684-0385-5

To Marsha and Rachel

Preface

Statistical performance evaluation has assumed an increasing amount of importance as we seek to design more and more sophisticated communication and information processing systems. The ability to predict a proposed system's performance without actually having to construct it is an extremely cost effective design tool.

This book is meant to be a first year graduate level introduction to the field of statistical performance evaluation. As such, it covers queueing theory (chapters 1-4) and stochastic Petri networks (chapter 5). There is a short appendix at the end of the book which reviews basic probability theory. At Stony Brook, this material would be covered in the second half of a two course sequence (the first half is a computer networks course using a text such as Schwartz's *Telecommunications Networks*). Students seem to be encouraged to pursue the analytical material of this book if they first have some idea of the potential applications.

I am grateful to B.L. Bodnar, J. Blake, J.S. Emer, M. Garrett, W. Hagen, Y.C. Jenq, M. Karol, J.F. Kurose, S.-Q. Li, A.C. Liu, J. McKenna, H.T. Mouftah and W.G. Nichols, I.Y. Wang, the IEEE and Digital Equipment Corporation for allowing previously published material to appear in this book.

My appreciation of this material has been enhanced by interaction with my students in Stony Brook's Congestion and Delay in Communications Systems Course. I am grateful to S. Rappaport for encouraging me to teach this course. Thanks are due for editorial assistance to Z. Ruder, K. Dreyhaupt, J. Day and J. Abrahams of Springer-Verlag and to A. Lazar of Columbia University. Thanks are due to M. Gerla for reviewing the manuscript. This book benefited from the drawing ability of L. Koh. The graph of the Poisson distribution was made by J. Shor. This book would not have been possible without the use of computer facilities supervised by J. Murray, M. Lee and G. Liu. I would like to gratefully acknowledge the support, in the course of this work, of the National Science Foundation (grant no. NCR-8703689) and of the SDIO/IST and managed by the U.S. Office of Naval Research (grant no. N00014-85-K0610).

Finally, I would like to thank my wife and daughter for their encouragement and support during the course of this project.

Parts of this book have benefited from the influence of a number of earlier, classic works which no personal library of performance evaluation should be without. These recommended readings are:

Bruell, S.C. and Balbo, G., *Computational Algorithms for Closed Queueing Networks*, North-Holland, N.Y., 1980.

Cooper, R.B., *Introduction to Queueing Theory*, North-Holland, N.Y., 1981.

Gross, D. and Harris, C.M., *Fundamentals of Queueing Theory*, Wiley, New York, 1974, 1985.

Kelly, F.P., *Reversibility and Stochastic Networks*, John Wiley & Sons, New York, 1979.

Kleinrock, L., *Queueing Systems, Vol. I: Theory*, Wiley, New York, 1975.

Kobayashi, H., *Modeling and Analysis: An Introduction to System Performance Evaluation*, Addison-Wesley, Reading, Mass., 1978.

Schwartz, M., *Telecommunications Networks: Protocols, Modeling and Analysis*, Addison-Wesley, Reading, Mass., 1987.

A solution manual is available from the author for instructors who adopt this book for a course. Please send requests on university stationery.

T.G.R.
Dept. of Electrical Engineering,
SUNY at Stony Brook,
Stony Brook, N.Y. 11794,
May 1990

Table of Contents

Chapter 1: The Queueing Paradigm

1.1 Introduction

This is a book about statistical prediction. It tells the story of how the behavior of complex electronic systems can be predicted using pencil, paper, the poetry of mathematics and the number crunching ability of computers.

The types of prediction that we would like to make include:

→ How many terminals can one connect to a time sharing computer and still maintain a reasonable response time?

→ What percentage of calls will be blocked on the outgoing lines of a small business's telephone system? What improvement will result if extra lines are added?

→ What improvement in efficiency can one expect in adding a second processor to a computer system? Would it be better to spend the money on a second hard disc?

What is the best architecture for a space division packet switch?

Paradigms are fundamental models which abstract out the essential features of the system being studied. This book deals with two basic paradigms: the paradigm of the queue (chapters 1-4) and the paradigm of the Petri net (chapter 5).

1.2 Queueing Theory

The study of queueing is the study of waiting. Customers may wait on a line, planes may wait in a holding pattern, telephone calls may wait to get through an exchange, jobs may wait for the attention of the central processing unit in a computer and packets may wait in the buffer of a node in a computer network. All these examples have been studied using the mathematical theory of queues or queueing theory.

Queueing theory has its roots early in the twentieth century in the early studies of the Danish mathematician A.K. Erlang on telephone networks and in the creation of Markov models by the Russian mathematician A.A. Markov. Today it is widely used in a broad variety of applications. Moreover the theory of queueing continues to evolve.

Before proceeding to the actual study of queueing networks in chapters 2,3 and 4, we will spend some time discussing models of queueing.

1.3 Queueing Models

What' s in a Name?

In the operations research literature the items moving through a queueing system are often referred to as "customers" as real human waiting line may be envisioned. When a computer scientist writes about queues, he or she will often refer to the items moving through the queueing system as "jobs" since the application is often jobs circulating through a computer system. A telephone engineer, meanwhile, will speak of "calls". Finally, in computer networks, packets of data pass through the nodes and lines of such networks so a queueing model will be phrased in terms of these "packets". While "customer", "job", "call" and "packet" all are meaningful, depending on the application, we will generally use the generic term "customer" to refer to the items that move through a queueing system.

The Simplest System

The simplest queueing model involves a single queue. We can make use of a commonly used schematic representation in order to illustrate it. This is done in Figure 1.1.

In this queueing system customers enter from the left and exit at the right. The circle represents the "server". For instance, in a supermarket check out line the server would be the employee operating the cash register. The open rectangle in Figure 1.1 represents the waiting line, or queue, that builds up behind the server.

A word about terminology is in order. The system of Figure 1.1 can be referred to as a "queueing system" or as a "queue" even though one of its components is also called the "queue". The meaning will usually be clear from the context.

In chapters 2,3 and 4 we will make very specific assumptions regarding the statistics of arrivals to the queueing system and for the time it takes for the server to process a customer.

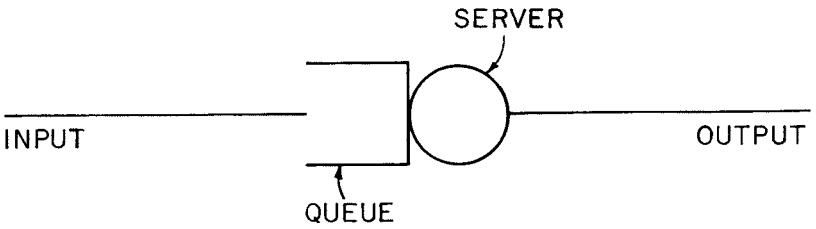

Fig. 1.1: A Single Server Queueing System.

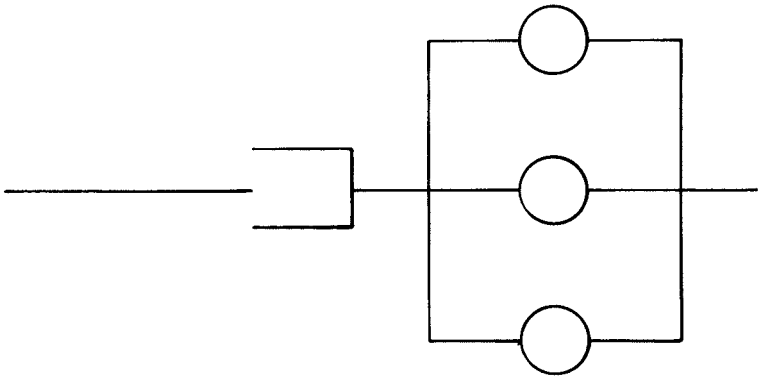

Fig. 1.2: An M/M/3 Queueing System

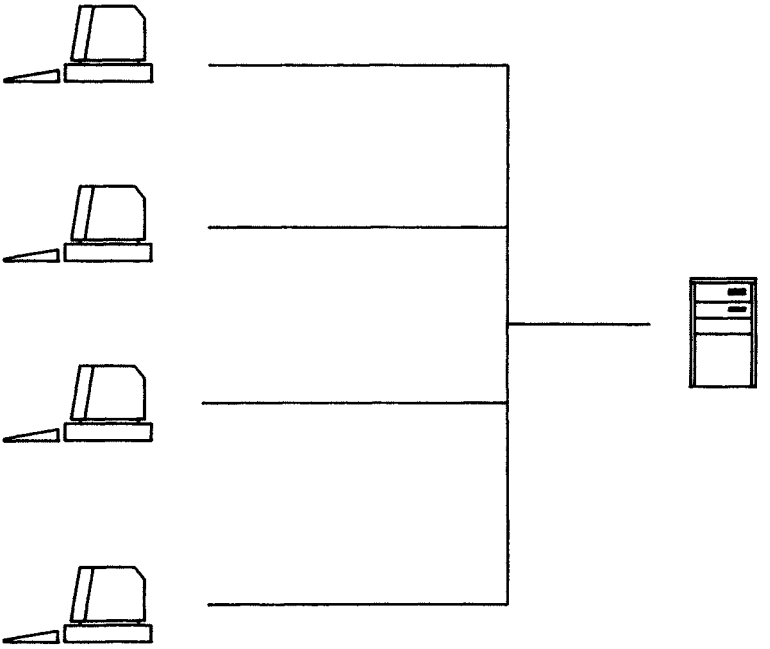

Fig. 1.3: Time Shared Computer & Terminals

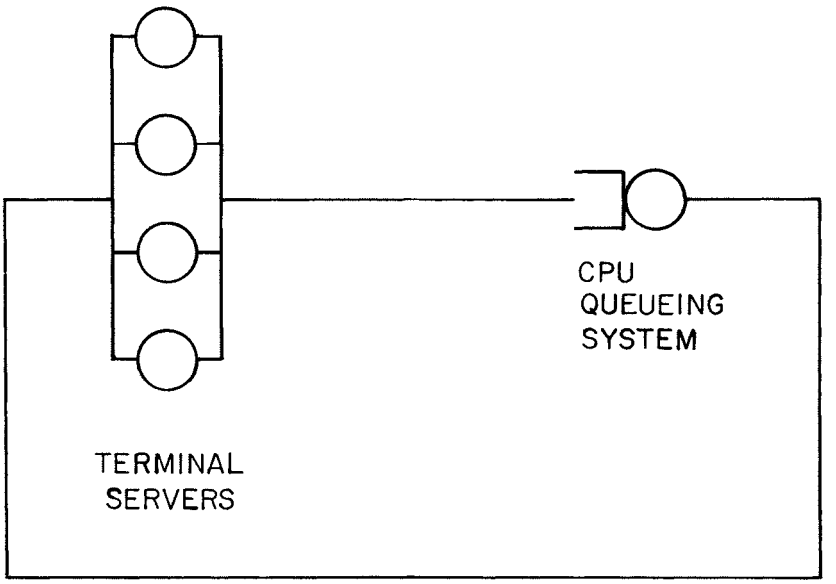

Fig. 1.4: Queueing Model of Time Shared Computer

Shorthand Notation

There is a standard shorthand notation used to describe queueing systems containing a single queue. As an example, consider the M/M/m/n queueing system. The characters refers to, respectively, the statistics of the arrival process, the statistics of the server, the number of servers and the number of customers that the queueing system can hold.

For instance, an M/M/3 queueing system is illustrated in Figure 1.2. This system might be used to model three tellers at a bank with a single common waiting line. There are three servers in parallel. Once a customer leaves a server, the customer at the head of the waiting line immediately takes his place at the server. There is no limit on the number of customers in such a queueing system as the fourth descriptor is absent from "M/M/3".

The characters in the first two positions indicate:

M: Markovian statistics

D: Deterministic Timing

G: General (arbitrary) statistics

Networks of Queueus

Many practical systems can be modeled as networks of queues. An "open" queueing network accepts and loses customers from/to the outside world. Thus the total number of customers in an open network varies with time. A "closed" network does not connect with the outside world and has a constant number of customers circulating throughout it.

For instance, in Figure 1.3 we have four terminals and a time shared computer. An abstracted, closed, queueing model of this system appears in Figure 1.4. Each terminal is modeled by a single server that holds at most one customer. A "customer" that circulates in this queueing network represents the control of the computer terminal. That is, the presence of a customer at a terminal server indicates that the person at the terminal is thinking or typing and the presence of a customer at the queue modeling the computer indicates that the person has hit "return" and is waiting for the prompt (exit from the computer queue). Naturally, in this specific queueing model a customer leaving a terminal server must eventually return to the same server.

Service Discipline

Most everyday queueing systems that we are use to serve customers in the order that they arrive. This is a First In First Out (FIFO) service discipline.

Other service disciplines are sometimes encountered in practical systems. In the Last In First Out (LIFO) service discipline the most recent arrival enters the server, either displacing the customer in service or waiting for it to finish. When service is resumed for the displaced customer at the point where service was disrupted, we have the LIFO service discipline with preemptive resume (LIFOPR). The LIFO service discipline could be useful, for instance, for modeling an emergency message system where recent messages receive priority treatment.

In the processor sharing (PS) discipline the processor (server) has a fixed rate of service it can provide that is equally distributed among the customers in the system. That is, there is no waiting queue, with each customer immediately receiving some service. Of course if there are many customers in the system, each receives a small fraction of the total processing effort. The PS discipline is useful for modeling multiprogramming in computers where the processor splits its effort among several jobs.

State Description

In the simplest state description, the state of a queueing network is a vector indicating the total number of customers in each queue at a particular time instant. Markovian statistics are often used in modeling queues as this state description is sufficient to completely describe a queueing network at an instant of time. That is, it is not necessary to include in the state description elapsed times in service or the time since the last arrival.

If there are several "classes" of customers the state description can be more detailed. Class may be used to decide arrival or service statistics by class or routing by class. One may now indicate the number of each class in each queue in the state description. At a more detailed level one may indicate the class of each customer at each waiting line position in the queue.

Unfortunately, as we shall see, even small queueing networks may have a number of states so large as to produce computational problems. A great deal of cleverness has gone into devising techniques to overcome this difficulty.

The Rest of This Book

In chapter 2 we will look at single queueing systems. Chapter 3 discusses networks of queues. Chapter 4 explains numerical algorithms for queueing networks. Finally, chapter 5 discusses the rather different and more recent paradigm of the Petri Net.

We conclude this chapter with two case studies of practical modeling drawn from the technical literature. They are included to show that the study and use of queueing models is a most practical enterprise.

1.4 Case Study I: Performance Model of a Distributed File Service

The following section discusses the performance modeling of a distributed file service. It specifically deals with the Digital Equipment Corp. VAX Distributed File Service (DFS). The file service provides remote file access for VAX/VMS systems. To a client DFS simply appears to be a local file service. The following is excerpted from "Design and Implementation of the VAX Distributed File Service" by William G. Nichols and Joel S. Emer in the Digital Technical Journal, No. 9, June 1989, Copyright, 1989, Digital Equipment Corporation (VAX is a DEC trademark).

Performance Analysis Model

For contrasting design alternatives, it is too time consuming to build many systems with different designs. Even for a given design, large multiple-client testing is very time consuming. Therefore, we developed a queueing network model to assess the performance of the file design alternatives quickly and quantitatively. The model represents a distributed system comprising the file server and multiple single-user workstations as client machines. Since we were primarily interested in the performance of the file server, the queueing network model and the workload we describe here represent multiple clients making requests only to a single DFS server.

Some important characteristics of the DFS communication mechanism affect the model of the system. The DFS protocols are connection oriented, with long-living connections between client and server machines. Also, these protocols generally have a strict request-response nature, so that at any time a client will usually have only one outstanding request to a server. As a result, the number of requests outstanding in the overall system (that is, requests being processed at the server or being processed or generated at the client) corresponds to the number of single-user client workstations in the system. Because of this characteristic, it is convenient to use a closed queueing network model to represent a distributed system in which multiple clients request service from a DFS server. In this model, the number of customers in the queueing network corresponds to the number of workstations.

Client requests to the server may experience queueing at each of the individual resources at the server node. The delays caused by queueing reduce the system performance as seen by the user. Therefore, we represent the server in substantial detail; each service center that represents a resource at the server also has a queue for requests that may await service at that resource. The server resources being modeled are:

- The network interface (The transmit and receive portions are each modeled separately.)

- The CPU
-The disk subsystem at the server

Since no queueing occurs at the client, the various segments of processing taking place at the client are represented by delay servers, i.e., service centers with no associated queue. A simple view of the distributed system is shown in Figure 1.5.

A particular system design and workload may be modeled by substituting the appropriate parameters into the model. For determining many of the system design parameters, we use a prototype file service that was developed in corporate research [RAMA 86].

The prototype had already demonstrated the feasibility of a VMS file service and some of its performance potential. This prototype itself represented only one design alternative. To estimate the other design alternatives, we used the prototype to provide parameters for our models.

All the models have the same set of service centers, but the centers are distributed differently in the various models. Therefore, we measured the time taken to execute the parts of the prototype that correspond to the service centers in our model. We could then analyze the models with real parameters to predict the performance of various design alternatives.

We constructed an artificial workload to drive our model. The workload was based on our understanding of typical user behavior and was also derived from measurements of typical timesharing environments [JAIN], [OUST]. A user at a client workstation was considered to invoke program images repeatedly. The programs invoked by the user require access to some number of files. The programs alternate between performing local computation and doing system service operations for access to these files. The files are assumed to be stored at the file server.

The amount of data transferred and the processing done at the server for each request depends on the type of request. The model distinguishes two types of request: control operations and data access operations. Control operations are operations such as open and close file which have a high computational component. On the other hand, data access operations are simple reads and writes. Data access operations usually have low computational requirements but require larger data transfers.

In between the program invocations, we assumed that the user spends a certain amount of time thinking or doing processing unrelated to our investigation. All processing at the client is represented in the model as delay servers with the appropriate branching probabilities.

To make the model tractable, we assumed that the service time at each service center is exponentially distributed. Since we are interested primarily in a file server's mean performance, this was an acceptable assumption [LAZO]. Even given this assumption, the model had to distinguish control and data access operations.

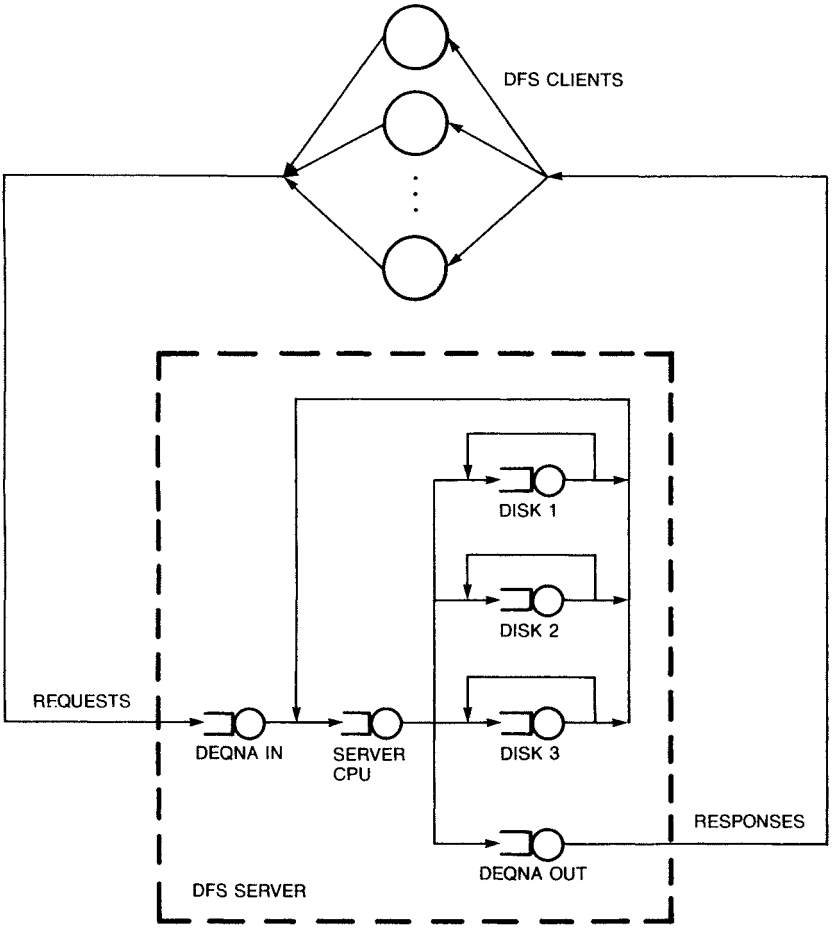

Fig. 1.5: Distributed File Service Queueing Model

Copyright 1989 DEC

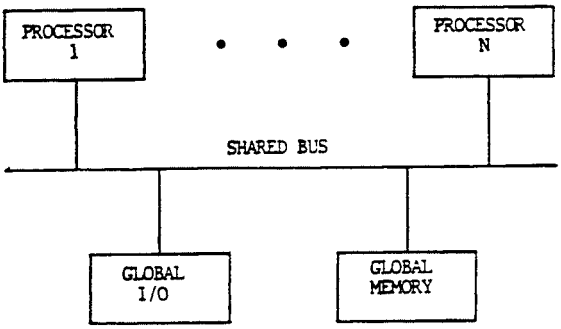

Fig. 1.6: Multiprocessor System
Copyright 1989 IEEE

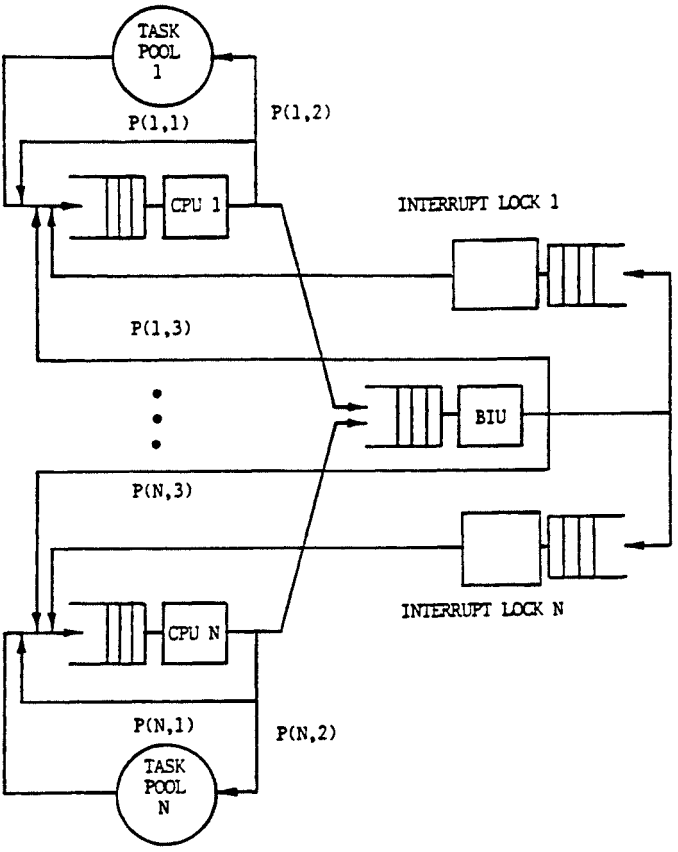

Fig. 1.7: Queueing Model of Multiprocessor
Copyright 1989 IEEE

Unfortunately, these two classes of operations have different service times at the various service centers. This difference means that the queueing network model does not satisfy the requirements for a product-form solution and may not be solved exactly [BASK 75]. We used a simple approximate-solution technique to solve this closed queueing network model. This technique is an extension of the basic multiclass mean value analysis technique [BRUE 80], [REIS 79]. The model characteristics, parameters, and the solution technique are described in greater detail in reference [RAMA 89]. Additional information appears in [EMER].

Case Study II: Single-Bus Multiprocessor Modeling

The following is a description of a model of a single bus multiprocessor. That is, in the system in question processors and global memory communicate amongst each other through a common bus. It originally appeared in "Modeling and Performance Analysis of Single-Bus Tightly-Coupled Multiprocessors" by B.L. Bodnar (AT&T Bell Laboratories) and A.C. Liu (Feng Chia University, Taiwan) in the IEEE Transactions on Computers, Vol. 38, No. 3, March 1989, copyright 1989 IEEE.

Description of the Multiprocessor Model

A tightly-coupled multiprocessor (TCMP) is defined as a distributed computer system where all the processors communicate through a single (global) shared memory. A typical physical layout for such a computer structure is shown in Figure 1.6. Figure 1.7 illustrates our queueing model for this (SBTCMP) architecture.

Each PE (with identifying index "i") is modeled as a finite set of tasks (the task pool), a CPU, a bus interface unit (BIU) which allows the PE to access the shared bus, and a set of queues associated with the CPU and BIU. Each CPU and BIU has mean service rate $\mu(i,1)$ and $\mu(i,2)$, respectively. Tasks are assumed to have a mean sleep time in the task pool of $\mu(i,0)^{-1}$ time units. We also assume the CPU and BIU operate independently of each other, and that all the BIU's in the multiprocessor can be lumped together into a single "equivalent BIU".

The branching probabilities are p(i,1), p(i,2) and p(i,3). The branching probability p(i,3) is interpreted as the probability that a task associated with PE i will join the CPU queue at PE i after using the BIU; 1-p(i,3) is then the probability that a task associated with PE i will wait for an interrupt acknowledgement at PE i. It is *not* the probability that a task will migrate from one PE to another. The problem of task migration is treated later in this paper.

Interrupts to the PE are assumed to occur at a mean rate $\mu(i,3)$. In general, the interrupt rate to PE i will be a function of task location, inter-task communication probability, speeds of resources on other PE's, global I/O devices, etc. The operation of the multiprocessor is now described. In this discussion, we will assume that the workload is fixed.

A task is assumed initially asleep in the task pool. When it awakens, it will queue for CPU usage in the ready list (the "ready list" is a list of tasks that are eligible to use the CPU as a result of either being awakened or as a result of having finished using some resource). Interrupt-driven jobs *are not* placed on the ready list; instead, they have their own (higher priority) queue. After using the CPU, the task may request more CPU service (in which case it will requeue in the ready list), it may go back to sleep in the task pool, or it may request bus usage.

If the task requests bus usage, it will enter the BIU queue. If the bus is free, the task will begin using the bus; otherwise, it will wait until the bus is released. After using the bus, the task will either request CPU time (it will be placed at the back of the ready list) or it will wait for an interrupt (i.e. an acknowledgement from a task residing on another PE or an acknowledgement from a globally accessible I/O device). *We assume that only one of the preceding events can occur at a given moment.* That is, CPU and interrupt processes can not occur concurrently. We also assume that a task queued for bus access will not continue using the CPU.

Tasks waiting for an interrupt or undergoing interrupt servicing will be referred to as "interrupt-driven tasks". Tasks waiting for interrupts are modeled by a queue feeding a "lock". The lock is drawn using a triangle to signify that it is enabled via an external stimulus. The purpose of the lock is to allow only one task to pass by it in response to an interrupt.

Upon receipt of an interrupt, an interrupt-driven task will usurp the CPU from *any other task.* If the task which was forced to release the CPU was an interrupt-driven task, then this preempted task will become the first entry on a last-come-first-served queue (i.e., a stack). If the preempted task was not an interrupt driven task, then it will become the first entry on the ready list.

We assume that *all* interrupt-driven tasks on the stack must be serviced before any noninterrupt-driven ones will obtain service. We will further assume that a preempted task can be restarted at the point where it was preempted. Hence, the CPU operates under two distinct service modes; namely, a low priority mode (such as round-robin, first-come-first-served, etc.) and a high priority last-come-first-served preemptive resume (LCFSPR).

A task residing in a given processor may also migrate to another processor. This could be done, for instance, in order to equalize load on the multiprocessor. Task migration from one PE to another has to be via the single time-shared bus (as this is the only communication medium). Note: The task migration problem is covered in Section IV of this paper.

In summary, this model takes into account the generalized overall behavior of the workload present on a single-bus tightly coupled multiprocessor. Specifically, it not only considers the bus contention caused by multiple processors attempting to access the shared memory, but also attempts to realistically include the local behavior of the various tasks running on these same processors.

Chapter 2: Single Queueing Systems

2.1 Introduction

In this chapter we will look at the case of the single queue. Even though chapters 3 and 4 will discuss networks of queues, it is probably safe to say that the majority of modeling work that has been done for queues has involved a single queue. One reason is that for any investigation into queueing theory the single queueing system is a natural starting point. Another reason is tractability: much more can be easily ascertained about single queues than about networks of queues.

We will start by examining the background of the most basic of queueing systems, the M/M/1 queue. This "Markovian" queue is distinguished by a Poisson arrival process and exponential service times. It is also distinguished by a one dimensional, and infinite in extent, state transition diagram. The coordinate of the one dimensional state transition diagram is simply the number of customers in the queueing system.

Later, by modifying the rates associated with this diagram and the diagram's extent we will arrive at a number of other useful queueing systems such as ones with multiple servers and ones with finite waiting line capacity.

We will spend some time looking at a non-Markovian queue of great interest, the M/G/1 queueing system. For this generalized queueing system a surprising amount of analytic information can be obtained.

Finally, for some queueing systems it is easier to calculate state probabilities in the z domain rather than directly. We will look at this z-transform technique in terms of two examples, the transient analysis of the M/M/1 queueing system and the M/G/1 queueing system.

2.2 The M/M/1 Queueing System

Consider the queueing system of Figure 2.1. It is the queueing system from which all queueing theory proceeds. This M/M/1 queueing system has two crucial assumptions. One is that the arrival process is a Poisson process and the other is that the single server has a service time which is an exponentially distributed random variable. These assumptions lead to a very tractable model and are reasonable for a wide variety of situations.

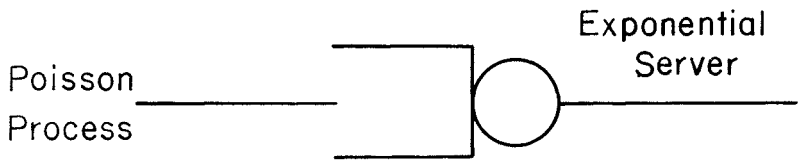

Fig. 2.1: M/M/1 Queueing System

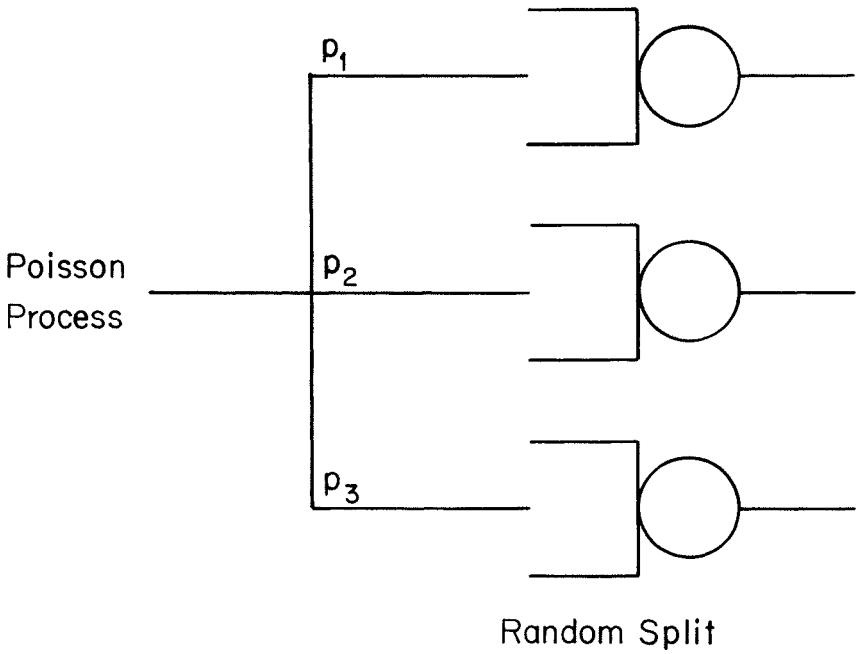

Fig. 2.2: Random Split of a Poisson Process

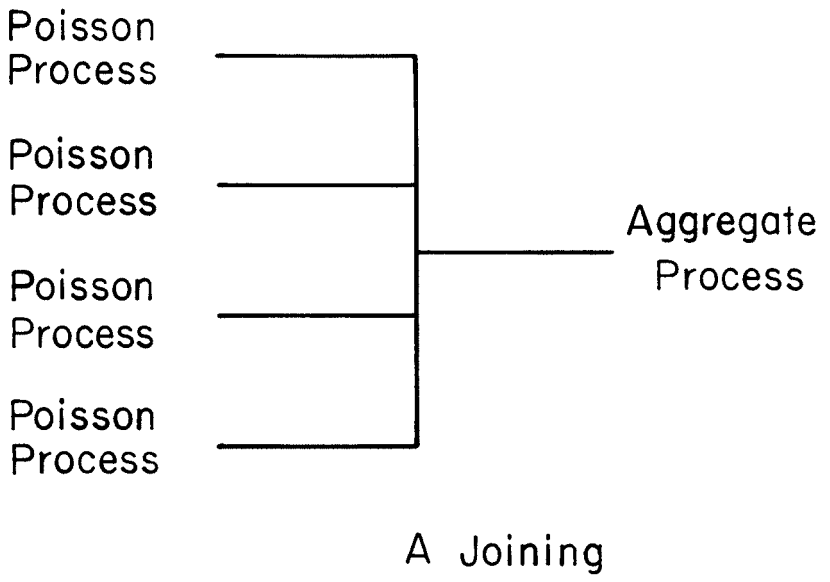

Fig. 2.3: Joining of Poisson Processes

We will start by examining these assumptions.

2.2.1 The Poisson Process

The *Poisson process* is the most basic of arrival processes. Interestingly, it is a purely random arrival process. One way of looking at it goes like this. Suppose that the time axis is divided into a large number of small time segments of width Δt. We will let the probability of a single customer arriving in a segment be proportional to the length of the segment, Δt, with a proportionality constant, λ, which represents the mean arrival rate.

$$P(\textit{exactly 1 arrival in } [t,t+\Delta t]) = \lambda \Delta t \tag{2.1}$$

$$P(\textit{no arrivals in } [t,t+\Delta t]) = 1 - \lambda \Delta t \tag{2.2}$$

$$P(\textit{more than 1 arrival in } [t,t+\Delta t]) = 0 \tag{2.3}$$

Here we ignore higher order terms in Δt.

Now we can make an analogy between the arrival process and coin flipping. Each segment is like a coin flip with $\lambda \Delta t$ being the probability of an arrival (say heads) and $1 - \lambda \Delta t$ being the probability of no arrival (say tails).

As $\Delta t \rightarrow 0$ we form the continuous time Poisson process. From the coin flipping analogy one can see that arrivals are independent of one another as they can be thought of as simply the positive results of a very large number of independent coin flips. Moreover one can see that no one instant of time is any more or less likely to have an arrival than any other instant.

Two ideas which extend the usefulness of the Poisson process are the concept of a random split of a Poisson process and a joining of Poisson processes. Figure 2.2 illustrates a queueing system with a random split. Arrivals are randomly split between three queues with independent probabilities p_1, p_2 and p_3. A situation involving a joining of independent Poisson processes is illustrated in Figure 2.3. Arrivals from a number of independent Poisson processes are joined or merged to form an aggregate process.

It turns out that random splits of a Poisson process are Poisson and that a joining of independent Poisson processes is also Poisson. A little thought with the coin flipping analogy will show that this is true.

One of the original applications of the Poisson process in communications was to model the arrivals of calls to a telephone exchange. The use of each telephone, at least in a first analysis, can be modeled as a Poisson process. These are joined into the aggregate Poisson stream of calls arriving at the exchange. It is very reasonable to model the aggregate behavior of a large number of independent users in this manner. However, this model may not always be valid. For instance, if there are correlations between an individual users calls - say given that the user has made one call he/she is more likely to make a second - then the Poisson assumption is not valid. Another example is that of a heavily loaded exchange. Calls are blocked (busy signal) which results in repeated attempts by individual users to get through. This would also invalidate the Poisson assumption.

Thus the life of a modeler can become complicated but we will continue to discuss elegant models with which we can make analytical progress.

2.2.2 Foundation of the Poisson Process

We will derive the underlying differential equation of the Poisson process by using difference equation arguments and letting $\Delta t \rightarrow 0$. We will start by letting

$$P_n(t) \equiv P(\#\ Arrivals = n\ at\ time\ t\ ) \tag{2.4}$$

and let $p_{ij}(\Delta t)$ be the probability of going from i arrivals to j arrivals in a time interval of Δt seconds. The approach will be similar to that of [KLEI 75].

The number of arrivals is the "state" of the system. It contains all the information necessary to completely describe the system. We can write:

$$P_n(t+\Delta t) = P_n(t)p_{n,n}(\Delta t) + P_{n-1}(t)p_{n-1,n}(\Delta t) \tag{2.5}$$

Again, we have neglected higher order terms in Δt. What this equation says is that one can arrive at a situation with n customers at time $t + \Delta t$ from either having n or n-1 customers at time t. Notice that we still assume that Δt is small enough so only one customer at most may arrive during this interval.

For this system we have the state transition diagram of Figure 2.4. Here the circles represent the states of the system (number of arrivals) and the transition rate λ is associated with each transition. We need a special equation for the state 0 to complete our difference equation description:

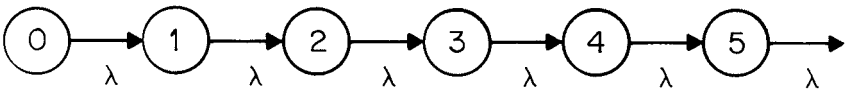

Fig. 2.4: Poisson Process State Transition Diagram

$$P_0(t+\Delta t) = P_0(t)p_{0,0}(\Delta t) \tag{2.6}$$

If we substitute in our previous expressions for the probability of exactly one arrival and the probability of no arrivals in an interval $[t,t+\Delta t]$ we have:

$$P_n(t+\Delta t) = P_n(t)(1 - \lambda\Delta t) + P_{n-1}(t)(\lambda\Delta t) \tag{2.7}$$

$$P_0(t+\Delta t) = P_0(t)(1 - \lambda\Delta t) \tag{2.8}$$

By multiplying out these expressions and re-arranging one can arrive at:

$$\frac{P_n(t+\Delta t) - P_n(t)}{\Delta t} = -\lambda P_n(t) + \lambda P_{n-1}(t) \tag{2.9}$$

$$\frac{P_0(t+\Delta t) - P_0(t)}{\Delta t} = -\lambda P_0(t) \tag{2.10}$$

If we let $\Delta t \to 0$ the set of difference equations becomes a set of differential equations:

$$\boxed{\begin{aligned} \frac{dP_n(t)}{dt} &= -\lambda P_n(t) + \lambda P_{n-1}(t) \\ \frac{dP_0(t)}{dt} &= -\lambda P_0(t) \end{aligned}} \tag{2.11}$$

Here $n \geq 1$. We now naturally wish to find the solution to these differential equations, $P_n(t)$. For the second equation, a knowledge of differential equations enables us to see that:

$$P_0(t) = e^{-\lambda t} \tag{2.12}$$

This makes the next equation:

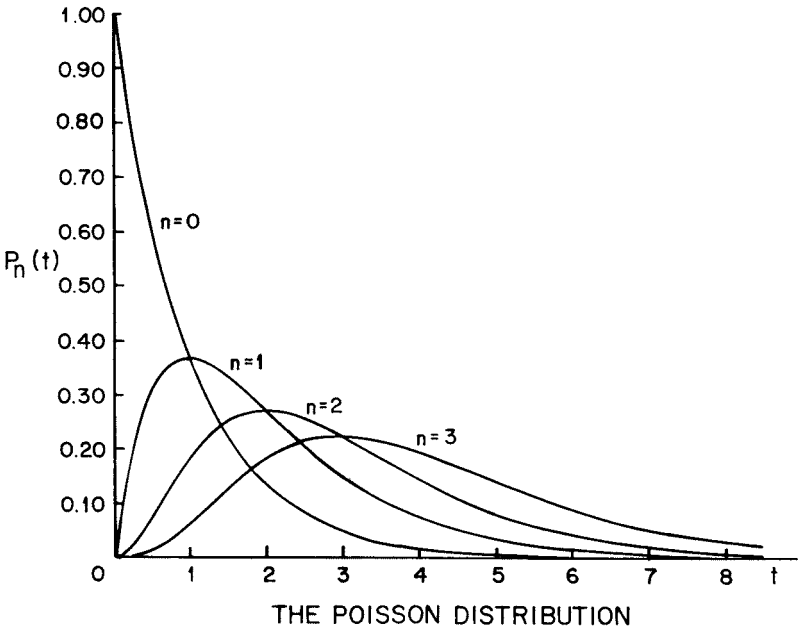

Fig. 2.5: Poisson Distribution

$$\frac{dP_1(t)}{dt} = -\lambda P_1(t) + \lambda e^{-\lambda t} \tag{2.13}$$

This equation has a solution of:

$$P_1(t) = \lambda t e^{-\lambda t} \tag{2.14}$$

This makes the next equation

$$\frac{dP_2(t)}{dt} = -\lambda P_2(t) + \lambda^2 t e^{-\lambda t} \tag{2.15}$$

which has a solution of:

$$P_2(t) = \frac{\lambda^2 t^2}{2} e^{-\lambda t} \tag{2.16}$$

Continuing, we would find by induction that:

$$\boxed{P_n(t) = \frac{(\lambda t)^n}{n!} e^{-\lambda t}} \tag{2.17}$$

This is the *Poisson distribution.* It tells us the probability of n arrivals in an interval of t seconds for a Poisson process of rate λ. A plot of the Poisson distribution appears in Figure 2.5.

Example: A telephone exchange receives 100 calls a minute on average, according to a Poisson process. What is the probability that no calls are received in an interval of five seconds?

The solution is simply:

$$P_0(t) = e^{-\lambda t} = e^{-100 \times \frac{1}{12}} = .00024$$

∇

2.2.3 Poisson Distribution Mean and Variance

We will calculate the mean and variance of the Poisson distribution. Let $\overline{n}$ be the mean number of arrivals in an interval of length t. Then:

$$\bar{n} = \sum_{n=0}^{\infty} nP_n(t) \tag{2.18}$$

$$\bar{n} = \sum_{n=0}^{\infty} n \frac{(\lambda t)^n}{n!} e^{-\lambda t} \tag{2.19}$$

$$\bar{n} = e^{-\lambda t} \sum_{n=1}^{\infty} \frac{(\lambda t)^n}{(n-1)!} \tag{2.20}$$

With a change of variables this is:

$$\bar{n} = e^{-\lambda t} \sum_{n=0}^{\infty} \frac{(\lambda t)^{n+1}}{n!} \tag{2.21}$$

$$\bar{n} = e^{-\lambda t} \lambda t \sum_{n=0}^{\infty} \frac{(\lambda t)^n}{n!} = e^{-\lambda t} \lambda t e^{\lambda t} = \lambda t \tag{2.22}$$

The equation $\bar{n} = \lambda t$ makes intuitive sense. The mean number $\bar{n}$ is proportional to the length of the interval t with rate constant λ.

For the variance, following the treatment in [THOM] and [KLEI 75] we will first calculate:

$$E(n(n-1)) = \sum_{n=0}^{\infty} n(n-1) \frac{(\lambda t)^n}{n!} e^{-\lambda t} \tag{2.23}$$

$$E(n(n-1)) = \sum_{n=2}^{\infty} \frac{(\lambda t)^n}{(n-2)!} e^{-\lambda t} \tag{2.24}$$

Making a change of variables:

$$E(n(n-1)) = e^{-\lambda t} (\lambda t)^2 \sum_{n=0}^{\infty} \frac{(\lambda t)^n}{n!} \tag{2.25}$$

$$E(n(n-1)) = e^{-\lambda t} (\lambda t)^2 e^{\lambda t} = (\lambda t)^2 \tag{2.26}$$

Now

$$E(n(n-1)) = E(n^2) - E(n) = (\lambda t)^2 \tag{2.27}$$

or

$$E(n^2) = (\lambda t)^2 + \lambda t \tag{2.28}$$

We also make use of the fact that the variance of n is:

$$\sigma_n{}^2 = E(n - \bar{n})^2 = E(n^2) + E(\bar{n})^2 - 2\bar{n}\,E(n) \tag{2.29}$$

$$\sigma_n{}^2 = E(n^2) - \bar{n}^2 \tag{2.30}$$

Substituting one has:

$$\sigma_n{}^2 = (\lambda t)^2 + \lambda t - (\lambda t)^2 = \lambda t \tag{2.31}$$

So the mean and the variance of the Poisson distribution are both equal to λt.

2.2.4 The Inter-Arrival Times

The times between successive events in an arrival process are called the inter-arrival times. For the Poisson process these times are independent exponentially distributed random variables. To see that this is true, we can write the probability that the time between arrivals is $\leq t$ as:

$$P(\textit{time btwn arrivals} \leq t) = 1 - P(\textit{time btwn arrivals} > t)$$

$$P(\textit{time btwn arrivals} \leq t) = 1 - P_0(t)$$

$$P(\textit{time btwn arrivals} \leq t) = 1 - e^{-\lambda t}$$

If we differentiate this we have:

$$\textit{Inter-arrival Density}\ (t) = \lambda e^{-\lambda t}$$

An exponential random variable is the only continuous random variable that has the *memoryless property.* Loosely speaking, this means that the previous history of the random variable is not needed in predicting the future. To be more specific, a complete state description of an M/M/1

queueing system at an instant of time consists only of the number of customers in the system. We do not have to include in the state description the time since the last arrival because the M/M/1 queueing system has a Poisson arrival process with exponential inter-arrival times. Furthermore we also do not have to include the time since the last service completion in the state description since the service time in an M/M/1 queueing system is exponentially distributed.

In terms of the Poisson arrival process the memoryless property means that the distribution of time until the next event is the same no matter what point in time we are at. Let us give an intuitive analogy. Suppose that trains arrive at a station according to a Poisson process with a mean time between trains of 20 minutes. Suppose now that you arrive at the station and are told by someone standing at the platform that the last train arrived nineteen minutes ago. What is the expected time until the next train arrives?

One would naturally like to answer one minute and this would be true if trains arrived deterministically exactly twenty minutes apart. However we are dealing with a Poisson arrival process. The answer is twenty minutes!

To see that this is true, we have to go back to our coin flipping explanation of the Poisson process. An event is the result of a positive outcome of a "coin flip" in an infinitesimally small interval. As time proceeds there are a very large number of such flips. When we put the Poisson process in this context it is easy to see that the distribution until the next positive outcome does not depend at all on when the last positive outcome occurred. That is, you can't predict when you'll flip heads by knowing the last time when heads occurred.

We can make this idea more precise. If an arrival occurs at time t=0 we know from before that the distribution of time until the next arrival is $\lambda e^{-\lambda t} \quad t \geq 0$ and the cumulative distribution function is $1-e^{-\lambda t} \quad t \geq 0$. Suppose now that no arrivals occur up until time t_0. Given this information what is the distribution of time until the next arrival? We can write it in the following way [KLEI 75] with the dummy variable t*:

$$P(t^* \leq t+t_0 \mid t^* > t_0) = \frac{P(t_0 < t^* \leq t+t_0)}{P(t^* > t_0)} \tag{2.32}$$

$$P(t^* \leq t+t_0 \mid t^* > t_0) = \frac{\int_{t_0}^{t+t_0} \lambda e^{-\lambda t^*} dt^*}{\int_{t_0}^{\infty} \lambda e^{-\lambda t^*} dt^*} \tag{2.33}$$

$$P(t^* \leq t+t_0 \mid t^* > t_0) = \frac{-e^{-\lambda t^*}\Big|_{t_0}^{t+t_0}}{-e^{-\lambda t^*}\Big|_{t_0}^{\infty}} \tag{2.34}$$

$$P(t^* \leq t+t_0 \mid t^* > t_0) = \frac{-e^{-\lambda(t+t_0)} + e^{-\lambda t_0}}{-0 + e^{-\lambda t_0}} \tag{2.35}$$

$$P(t^* \leq t+t_0 \mid t^* > t_0) = 1 - e^{-\lambda t} \tag{2.36}$$

This is the same cumulative distribution function we had when we asked what the distribution was at time t=0!

We close this section by noting the *discrete* distribution with the memoryless property is the geometric distribution [FELL].

2.2.5 The Markov Property

The memoryless property is more accurately referred to as the *Markov property.* It says, intuitively, that one can predict the future state of the system based on the present state as well as if one had the past history of the state available. In terms of a continuous, non-negative, random time T, this Markov property can be expressed as [COOP]

$$P(T > t_0+t \mid T > t_0) = P(T > t) \tag{2.37}$$

for every $t_0 > 0$ and every $t > 0$. In terms of a discrete random variable $X(t_n)$ one can say [KLEI 75] that

$$P(X(t_{n+1})=x_{n+1} \mid X(t_n)=x_n, X(t_{n-1})=x_{n-1}, \ldots, X(t_1)=x_1) =$$

$$= P(X(t_{n+1})=x_{n+1} \mid X(t_n)=x_n) \tag{2.38}$$

where the t_i are monotonically increasing and the x_n take one some values from a discrete state space.

Markovian systems are memoryless systems. This makes them relatively simple to analyze since when one describes the state of the system one doesn't need to take into account the time since the last arrival or the time that service has been underway. When the arrival process is not Poisson or the service time is not exponential then we must take these quantities into account. This makes understanding the behavior of even a single

queue under these conditions a non-trivial undertaking.

A concept that we will come across later is that of a *Markov chain.* This is a Markov process with a discrete state space. The appearance of the state transition diagram of the M/M/1 queueing system which we are about to see (Figure 2.6) accounts for the use of the word "chain".

2.2.6 Exponential Service Times

In an M/M/1 queueing system the service times are independent exponentially distributed random variables. That is, they occur according to the probability density $\mu e^{-\mu t}$ where μ is the mean service rate and $\frac{1}{\mu}$ is the mean time for service. As has been discussed, this type of server is memoryless and thus produces a tractable analysis.

How realistic is an exponential server? It has been found to be reasonable for modeling the duration of telephone conversations. But it has also been used in situations where the physical basis for its use is weaker. For instance, the time to transmit fixed length packets of information in computer networks is often modeled by an exponential random variable. Sometimes this sort of assumption is made for reasons of tractability, even if some degree of realism is sacrificed. This assumption may not be too bad if all one is interested in are mean values of performance and not the higher moments.

The exponential server will be with us for a long time because, in conjunction with the Poisson process, it leads to Markovian systems and the associated elegant network wide results of the next two chapters.

2.2.7 Foundation of the M/M/1 Queueing System

Let us take Figure 2.4 and add transitions that represent servicing to create the state transition diagram of Figure 2.6. This state transition diagram represents what is called a *birth-death process.* That is, if we think of the state as being a population size, it may increase by one member at a time ("birth") or it may decrease by one member at a time ("death"). There are other models that may change by several members at a time. In the context of the performance evaluation of information systems the state of the system can actually be thought of as the number of packets in a communication processor, the number of new calls in a telephone exchange computer or the number of jobs in a computer. The approach we'll take is similar to that in [KLEI 75].

When we were discussing the Poisson process we had the events:

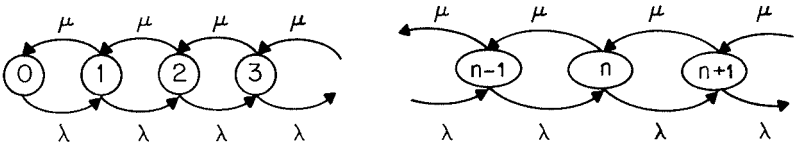

Fig. 2.6: M/M/1 System State Transition Diagram

$$P(\textit{exactly 1 arrival in } [t,t+\Delta t]) = \lambda \Delta t$$

$$P(\textit{no arrivals in } [t,t+\Delta t]) = 1 - \lambda \Delta t$$

We add to this two more events:

$$P(1 \textit{ service completion in } [t,t+\Delta t]) = \mu \Delta t$$

$$P(\textit{no service completions in } [t,t+\Delta t]) = 1 - \mu \Delta t$$

Also again we will define:

$$P_n(t) \equiv P(\# \textit{ Arrivals } = n \textit{ at time } t)$$

and let $p_{ij}(\Delta t)$ be the probability of going from i arrivals to j arrivals in a time interval of Δt seconds.

Whereas previously the state of the system was the number of Poisson arrivals, it is now the number in the queueing system, including the one in service. We can write:

$$P_n(t+\Delta t) = \tag{2.39}$$

$$P_n(t)p_{n,n}(\Delta t) + P_{n-1}(t)p_{n-1,n}(\Delta t) + P_{n+1}(t)p_{n+1,n}(\Delta t)$$

Higher order terms in Δt are neglected.

This equation says that one can arrive at a situation with n customers at time $t+\Delta t$ from either having n-1, n or n+1 customers at time t. Because this is a birth-death process, these are the only possibilities. For the state 0 we will need a special equation:

$$P_0(t+\Delta t) = P_0(t)p_{0,0}(\Delta t) + P_1(t)p_{1,0}(\Delta t) \tag{2.40}$$

We will now substitute the previous expressions for the probabilities of the various events in an interval $[t,t+\Delta t]$ into this equation to yield:

$$P_n(t+\Delta t) = \tag{2.41}$$

$$P_n(t)(1-\lambda\Delta t)(1-\mu\Delta t) + P_{n-1}(t)(\lambda\Delta t) + P_{n+1}(t)(\mu\Delta t)$$

$$P_0(t+\Delta t) = P_0(t)(1-\lambda\Delta t) + P_1(t)(\mu\Delta t)$$

By multiplying out these expressions, re-arranging and letting $\Delta t \to 0$ one has:

$$\frac{dP_n(t)}{dt} = -(\lambda+\mu)P_n(t) + \lambda P_{n-1}(t) + \mu P_{n+1}(t)$$

$$\frac{dP_0(t)}{dt} = -\lambda P_0(t) + \mu P_1(t) \qquad (2.42)$$

Here $n \geq 1$.

These differential equations describe the evolution in time of the state probabilities of the M/M/1 queueing system. But what do these equations really mean? To answer this question we will need:

2.2.8 Flows and Balancing

One of the most intuitively pleasing concepts in queueing theory is the way in which one can make an analogy between the flow of current in an electric circuit and the flows in the state transition diagram of a queueing system. But what is "flowing" in the state transition diagram?

The answer is that *probability flux* flows from one state to another along the transitions. The probability flux along a transition, numerically, is the product of the probability of the state at which the transition originates and the transition rate associated with the transition. The probability flux along a transition, physically, is the mean number of times per second that the event corresponding to the transition occurs. For instance, if a transition has a rate of 10 sec^{-1} and the probability of the state at which the transition originates is .1 then the transition is actually traversed by the system state once a second, on average.

The analogy with electric circuits is straight forward. The state probabilities correspond to nodal voltages and the transition rates correspond to conductance (inverse resistance) values. There are some differences though. Specifically:

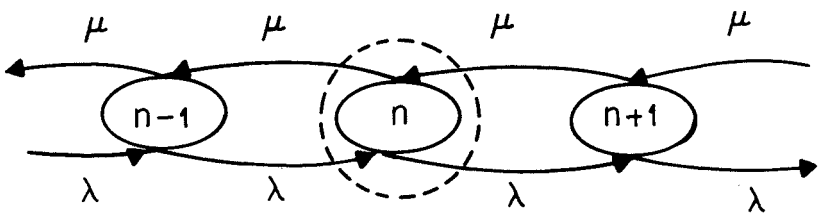

Fig. 2.7: Global Balance Boundary Around a State

There is no analog of a voltage source or battery in the queueing world. Rather we have the normalization equation that holds that at any instant of time the sum of the state probabilities sums to one. This normalization equation "energizes" the state transition diagram in the sense that it guarantees that there will be non-zero flows.

The direction of flow is preset in the state transition diagram by the transition direction. As has been mentioned what "flows" is the product of the probability of the state at which the transition originates and the transition rate. This is different from the electric circuit case where a branch current's magnitude and direction depend on the difference of voltages of the two nodes to which the branch is connected.

The most useful concept that carries over from the electric circuit world to the queueing theory world is the idea of looking at what flows across boundaries in the circuit or state transition diagram. For instance, suppose we draw a boundary around state n in the state transition diagram of the M/M/1 queueing system, as in Figure 2.7. What we are going to do is equate the mean number of times a second the state is entered with the mean number of times one leaves the state. What flows out of the state at time t is

$$-(\lambda + \mu)P_n(t)$$

and what flows into the state from states n-1 and n+1 is:

$$\lambda P_{n-1}(t) + \mu P_{n+1}(t)$$

The difference between these two quantities is just the change in the probability of state n or:

$$\frac{dP_n(t)}{dt} = -(\lambda+\mu)P_n(t) + \lambda P_{n-1}(t) + \mu P_{n+1}(t) \qquad (2.43)$$

But this is just the same differential equation that we found using difference equation arguments in the last section! The equation for $\frac{dP_0(t)}{dt}$ can be derived in a similar manner.

So for the M/M/1 queueing system we have a set of differential equations to describe the evolution of the state probabilities as a function of time. As with electric circuits, this is useful if one wants a *transient analysis* of the queueing system. This is the study of the queueing system

over a limited time period. For instance, we may be interested in the queueing system's behavior for the first ten minutes after service commences for an initially empty queue. Then the initial conditions of the set of differential equations becomes

$$P_0(0) = 1.0 \tag{2.44}$$

$$P_n(0) = 0.0 \qquad n \geq 1$$

and the differential equations can be solved. We will look at this situation in section 2.11.

An interesting thing happens when we let $\frac{dP_n(t)}{dt} = 0$. This is the situation that occurs when the queueing system has been running for a long time, transient behavior has settled out and the queue is in equilibrium. *Equilibrium or steady-state analysis* deals with this situation.

Since the probabilities do not change with time we can write:

$$0 = -(\lambda+\mu)p_n + \lambda p_{n-1} + \mu p_{n+1}$$

$$0 = -\lambda p_0 + \mu p_1 \qquad (2.45)$$

Here $n \geq 1$. Note that we have changed probability to lower case.

These equations are now a set of linear equations rather than a set of differential equations. Each equation can be arrived at by drawing a boundary completely around a specific state and equating what flows into the state with what flows out of the state. An equation arrived at in such a manner is known as a *global balance equation.* The reader familiar with electric circuits will be reminded of Kirchoff's current conservation law which holds that the *net* flow into and out of a circuit node equals zero.

We can calculate the equilibrium state probabilities of any queueing system with a finite number of states in the following manner. For each of the N states we write one global balance equation. This gives us a set of N linear equations with N unknowns. Any one of the equations is redundant and should be replaced with the equation

$$p_0 + p_1 + p_2 + \ldots\ldots + p_N = 1$$

which normalizes the solution. The resultant set of linear equations can then be solved using standard numerical techniques. The reason that we

have emphasized the calculation of the state probabilities is that many performance measures of interest are functions of the state probabilities.

The difficulty with this approach is that realistic queueing models often have extremely large state spaces making numerical solution computationally infeasible.

Example: In [KAUF] L. Kaufman, B. Gopinath and E.F. Wunderlich of AT&T Bell Laboratories present a Markovian model of packet switches. A packet switch is a dedicated processor which transmits, relays and receives packets of digital information. It requires four dimensions to describe each packet switch in this model:

$$h = \#\ of\ local\ packets\ in\ input\ buffer$$

$$i = \#\ of\ foreign\ packets\ in\ input\ buffer$$

$$j = \#\ of\ packets\ in\ output\ buffer$$

$$k = \#\ of\ outstanding\ acknowledgements$$

Here local packets are headed for local destinations while foreign packets are headed for foreign packet switches. In the two packet switch case that is considered in this paper the number of states is tabulated as follows (copyright 1981 IEEE):

Number of Model States			
Buffer Size	Window Size	6-D Model	8-D Model
4	2	961	4,225
5	2	2,116	12,321
6	2	4,096	30,625
6	3	5,476	38,025
8	4	21,025	211,600
9	4	34,225	416,025

Here the window size refers to the allowable number of outstanding acknowledgements. The growth in the number of states for the two packet switch, eight dimensional model is impressive given the relatively small buffer and window size. A reduced six dimensional model is possible if one doesn't distinguish between the number of local and foreign packets in each switch but only looks at the total number of packets in the input buffer. While this reduces the state space size, modest increases in the buffer and window size would negate this savings.

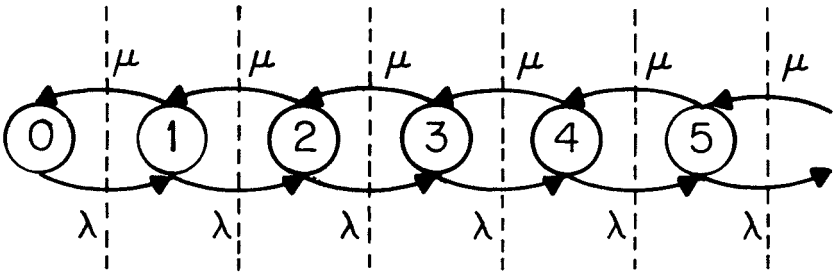

Fig. 2.8: Local Balance Boundary Between States

▽

Luckily, there is a large class of useful networks which satisfy a more specific type of balancing that leads to an analytic solution. Suppose we redraw the M/M/1 queueing system state transition diagram as in Figure 2.8. Here we've drawn vertical boundaries between each adjacent pair of states. Each boundary actually separates the state transition diagram into two halves. We can equate the flow from left to right across such a boundary with the flow from right to left. We thus obtain a set of *local balance equations*:

$$\lambda p_0 = \mu p_1 \tag{2.46}$$

$$\lambda p_1 = \mu p_2$$

$$\lambda p_2 = \mu p_3$$

$$\lambda p_{n-1} = \mu p_n$$

From these equations we can easily write the recursions:

$$p_1 = \frac{\lambda}{\mu} p_0 \tag{2.47}$$

$$p_2 = \frac{\lambda}{\mu} p_1$$

$$p_3 = \frac{\lambda}{\mu} p_2$$

$$p_n = \frac{\lambda}{\mu} p_{n-1}$$

Substituting one into each other leads to the equation:

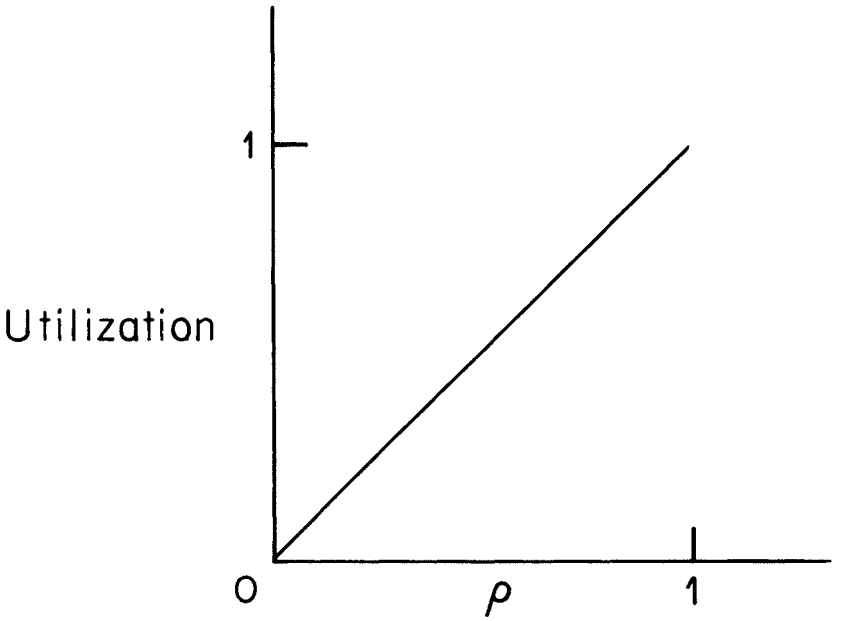

Fig. 2.9: M/M/1 System Utilization

$$p_n = \left(\frac{\lambda}{\mu}\right)^n p_0 \qquad (2.48)$$

$$p_0 = \frac{1}{\sum_{n=0}^{\infty}\left(\frac{\lambda}{\mu}\right)^n}$$

Here we used the normalization (conservation of probability) equation to solve for p_0. This result is the one dimensional version of what is known as the product form solution for the state probabilities. We will see the generalization in chapter 3.

2.2.9 The M/M/1 Queueing System in Detail

We can further simplify the expression for p_0 if we make use of the identity

$$\sum_{n=0}^{\infty} \rho^n = \frac{1}{1-\rho} \qquad 0 \leq \rho < 1 \qquad (2.49)$$

so that

$$p_n = \rho^n p_0 \qquad (2.50)$$

$$p_0 = (1-\rho) \qquad (2.51)$$

or

$$p_n = \rho^n (1-\rho) \qquad (2.52)$$

where we are letting $\rho = \lambda/\mu$. The variable ρ is known as *utilization*. This makes sense since for an M/M/1 queueing system the utilization of the queueing system is the probability that the queueing system is non-empty (Figure 2.9):

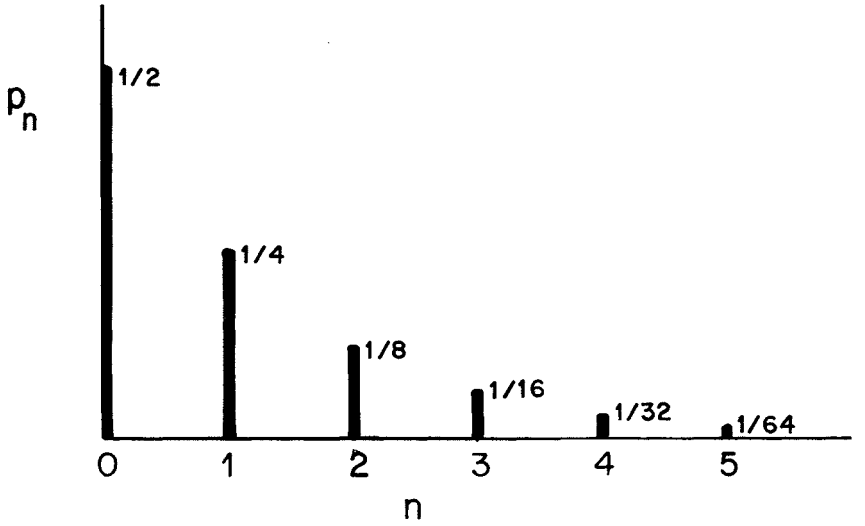

Fig. 2.10: p_n for M/M/1 system when $\rho=1/2$

$$U = 1 - p_0 = \rho \qquad (2.53)$$

The quantity ρ can also be viewed as the normalized offered load. That is, λ can vary from near zero (light load) to near, but not greater than, μ (heavy load). Thus ρ can vary between 0 and 1.

The state space of the M/M/1 queueing system is infinite in extent. Physically, we are saying that the queueing system has a potentially infinite sized waiting line. However since p_n is proportional to ρ^n the probability of the nth state is smaller than the probability of the n-1st state by a factor of ρ. Thus the probability of a larger waiting line is smaller than the probability of a smaller waiting line. This can be seen in Figure 2.10 where $\rho = 1/2$.

Example: Suppose that ρ equals either .1, .5, or .9. The table below illustrates the probabilities of different waiting line sizes:

Probs. of Queue System Occupancy			
	ρ=.1	ρ=.5	ρ=.9
$P(0\leq \ \leq 3)$	.9999	.9375	.3439
$P(4\leq \ \leq 7)$	1×10^{-4}	.0586	.2256
$P(8\leq \ \leq 11)$	1×10^{-8}	.00366	.1480
$P(\geq 12)$	1×10^{-12}	.000249	.2825

From the table we can see that for a lightly loaded queueing system (ρ=.1) there are usually less than four customers in the system. When ρ=.5 the probability that there are less than four customers drops to 94%. At ρ=.9 it is only 34% and there is a 28% chance of twelve or more customers.

∇

What we have implied in all of this and what we can state directly now is that the arrival rate can never be greater than the service rate for an M/M/1 queueing system. Put another way, ρ can never be greater than

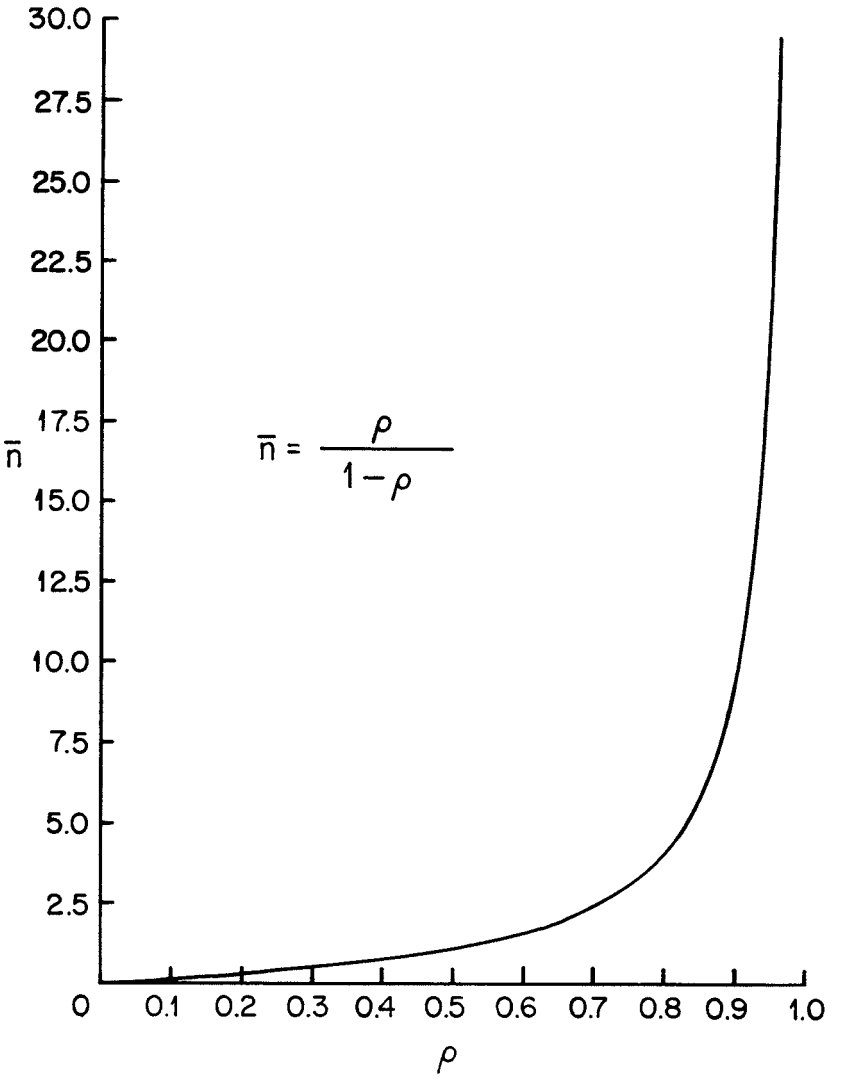

Fig. 2.11: M/M/1: Avg. Number of Customers

one since the queue size would be indefinitely increasing. That is, the queueing system would no longer be in equilibrium.

It is easy to calculate the average number in the queueing system [GROS] [KLEI 75] as follows. The average number is:

$$\bar{n} = E(n) = \sum_{n=0}^{\infty} np_n \tag{2.54}$$

$$\bar{n} = \sum_{n=0}^{\infty} n(1-\rho)\rho^n \tag{2.55}$$

$$\bar{n} = (1-\rho)\sum_{n=0}^{\infty} n\rho^n \tag{2.56}$$

But:

$$\sum_{n=0}^{\infty} n\rho^n = \rho \sum_{n=1}^{\infty} n\rho^{n-1} \tag{2.57}$$

$$\sum_{n=0}^{\infty} n\rho^n = \rho \frac{d}{d\rho} \sum_{n=0}^{\infty} \rho^n \tag{2.58}$$

$$\sum_{n=0}^{\infty} n\rho^n = \rho \frac{d}{d\rho} \frac{1}{1-\rho} \tag{2.59}$$

$$\sum_{n=0}^{\infty} n\rho^n = \frac{\rho}{(1-\rho)^2} \tag{2.60}$$

So by substituting:

$$\boxed{\bar{n} = \frac{\rho}{1-\rho}} \tag{2.61}$$

This function is plotted in Figure 2.11. What is most noticeable is the large non-linear increase in the waiting line size as $\rho \rightarrow 1$. Interestingly, this function is often used as a "penalty function" in computer network optimization algorithms to penalize large waiting line sizes [GERL].

We can also calculate the variance in this waiting line size. This is defined as:

$$\sigma_n^2 = \sum_{n=0}^{\infty} (n-\bar{n})^2 p_n \tag{2.62}$$

$$\sigma_n^2 = \sum_{n=0}^{\infty} n^2 p_n + \bar{n}^2 \sum_{n=0}^{\infty} p_n - 2\bar{n} \sum_{n=0}^{\infty} n p_n \tag{2.63}$$

$$\sigma_n^2 = \sum_{n=0}^{\infty} n^2 p_n + \bar{n}^2 - 2\bar{n}^2 \tag{2.64}$$

$$\sigma_n^2 = \sum_{n=0}^{\infty} n^2 p_n - \bar{n}^2 \tag{2.65}$$

But:

$$\sum_{n=0}^{\infty} n^2 p_n = \sum_{n=0}^{\infty} n^2 (1-\rho)\rho^n \tag{2.66}$$

$$\sum_{n=0}^{\infty} n^2 p_n = (1-\rho) \sum_{n=0}^{\infty} n^2 \rho^n \tag{2.67}$$

And:

$$\sum_{n=0}^{\infty} n^2 \rho^n = \frac{\rho(1+\rho)}{(1-\rho)^3} \tag{2.68}$$

So substituting yields:

$$\sigma_n^2 = \frac{\rho(1+\rho)}{(1-\rho)^2} - \frac{\rho^2}{(1-\rho)^2} \tag{2.69}$$

And:

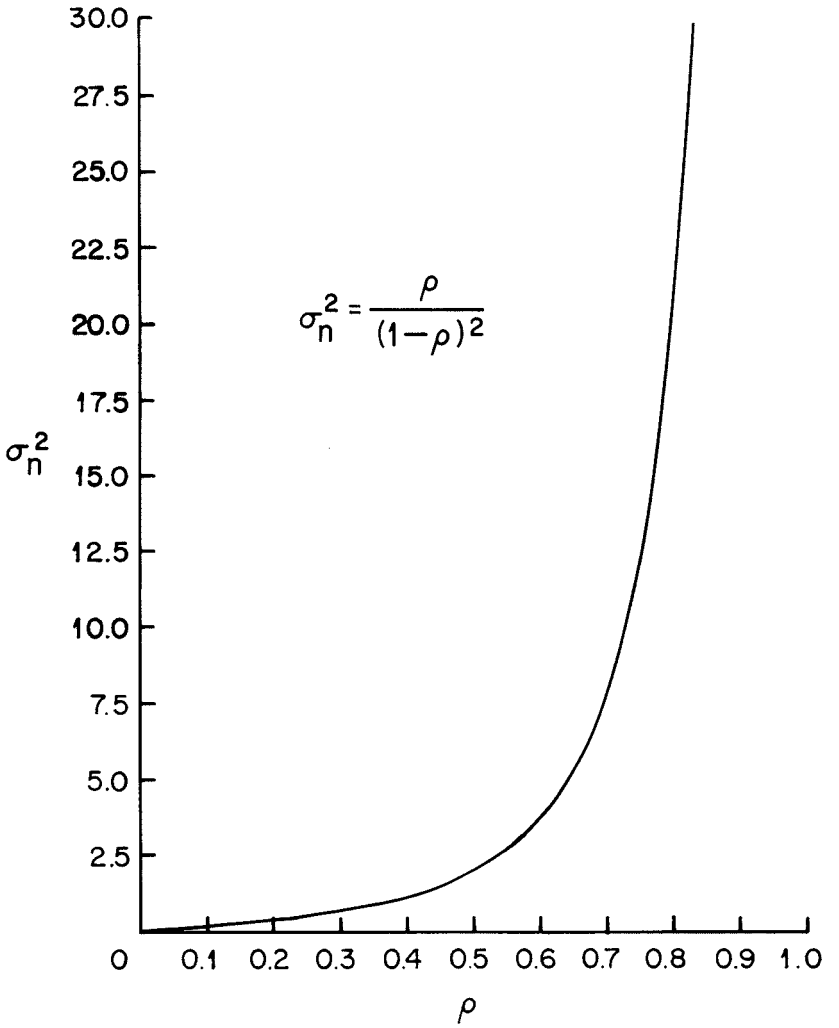

Fig. 2.12: M/M/1: Variance of Number of Customers

$$\sigma_n^2 = \frac{\rho}{(1-\rho)^2} \qquad (2.70)$$

This function is plotted in Figure 2.12. What we can see is that the variability of the number in the queueing system increases drastically as $\rho \rightarrow 1$.

2.3 Little's Law

We now take some time out from our discussion of specific queueing systems to look at a simple result that holds under a surprisingly large variety of situations. This is Little's Law. It relates the mean number of customers in a queueing system, $\overline{n}$; the mean arrival rate to the queueing system, λ; and the mean waiting time to get thru the queueing system, $\overline{\tau}$:

$$\overline{n} = \lambda \overline{\tau} \qquad (2.71)$$

Intuitively this equation makes sense. For example, if an assembly line accepts twenty chassis an hour and takes three hours to produce a finished product, then at any time there are sixty units in various states of assembly in the assembly line.

Proofs of this deceptively simple result can become quite involved. We will give three proofs of Little's Law:

Proof 1: We will first consider a proof, due to [GROS], for a queueing system with Poisson arrivals that is FIFO with a single server and a general service distribution. This is the M/G/1 queueing system. We start by noting that the probability that there are n customers in the system when a customer departs is equal to the probability that n customers arrive during the time spent in the queueing system by this customer. It turns out that the probability of n customers in the queueing system at a departure instant is equal to the probability of there being n customers in the queueing system at any time. We will return to this point in section 2.12.3. So:

$$p_n = \int_0^\infty Prob(n \text{ arrivals during } [0,T] \mid T=t) d\overline{\tau}(t) \qquad (2.72)$$

Here $\bar{\tau}(t)$ is the cumulative distribution function of waiting time. Using the Poisson distribution:

$$p_n = \int_0^\infty \frac{(\lambda t)^n}{n!} e^{-\lambda t}\, d\bar{\tau}(t) \tag{2.73}$$

Here $n \geq 0$. Now:

$$\bar{n} = E(n) = \sum_{n=0}^{\infty} n p_n \tag{2.74}$$

Substituting:

$$\bar{n} = \sum_{n=1}^{\infty} \frac{n}{n!} \int_0^\infty (\lambda t)^n e^{-\lambda t}\, d\bar{\tau}(t) \tag{2.75}$$

The integration and summation can be interchanged:

$$\bar{n} = \int_0^\infty d\bar{\tau}(t) \sum_{n=1}^{\infty} e^{-\lambda t} \frac{(\lambda t)^n}{(n-1)!} \tag{2.76}$$

$$\bar{n} = \int_0^\infty \lambda t e^{-\lambda t}\, d\bar{\tau}(t) \sum_{n=1}^{\infty} \frac{(\lambda t)^{n-1}}{(n-1)!} \tag{2.77}$$

$$\bar{n} = \int_0^\infty \lambda t e^{-\lambda t}\, d\bar{\tau}(t) \sum_{n=0}^{\infty} \frac{(\lambda t)^n}{n!} \tag{2.78}$$

$$\bar{n} = \int_0^\infty \lambda t e^{-\lambda t}\, d\bar{\tau}(t) e^{\lambda t} \tag{2.79}$$

$$\bar{n} = \lambda \int_0^\infty t\, d\bar{\tau}(t) \tag{2.80}$$

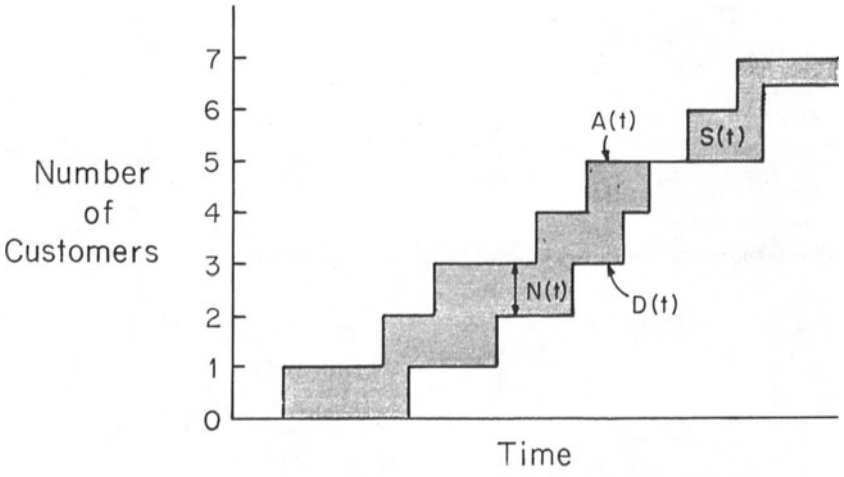

Fig. 2.13: Proving Little's Law

$$\boxed{\bar{n} = \lambda \bar{\tau} \quad (2.81)}$$

Proof 2: This is a heuristic proof due to P.J. Burke and related by Cooper [COOP]. We start by assuming that the mean values $\bar{n}$ and $\bar{\tau}$ exist and we have an equilibrium situation. We will look at a long interval $[0,t]$. The mean number of customers who arrive is λt. Let each customer bring his/her waiting time with them. The mean amount of waiting time brought in during the interval is $\lambda t \bar{\tau}$.

Now each customer in the queueing system "uses" up waiting time linearly with time. So if $\bar{n}$ is the mean number of customers present during the interval then $\bar{n}t$ is the mean amount of time used up during [0,t]. If we let $t \to \infty$ we can expect that the amount of waiting time brought into the queueing system must equal the amount of waiting time used up. Thus

$$\lim_{t \to \infty} \frac{\lambda t \bar{\tau}}{\bar{n} t} = 1 \qquad (2.82)$$

and so:

$$\boxed{\bar{n} = \lambda \bar{\tau} \quad (2.83)}$$

In this heuristic proof we made no special assumptions.

Proof 3: This popular type of proof is based on a diagram like that of Figure 2.13. Versions of it appear in [KOBA],[SCHW 87],[GROS] and [KLEI 75]. We will follow the approach in [KLEI 75]. As in proof 2, we will consider an interval [0,t]. In Figure 2.13 we have A(t) as the number of arrivals in [0,t] and D(t) as the number of departures. Then the number in the system is:

$$N(t) = A(t) - D(t) \qquad (2.84)$$

We will now present three simple relations amongst the quantities in Figure 2.13 which can be brought together to form Little's Law. The first is

$$\lambda_t = \frac{A(t)}{t} \qquad (2.85)$$

where λ_t is the mean arrival rate during [0,t]. The second is

$$\bar{\tau}_t = \frac{S(t)}{A(t)} \tag{2.86}$$

where $\bar{\tau}_t$ is the mean time spent in the queueing system by each customer and S(t) is the cumulative area between the curves A(t) and D(t) at time t. This area, at t, is the total time spent in the queueing system by all customers up to time t. Finally, the third equation is

$$\bar{n}_t = \frac{S(t)}{t} \tag{2.87}$$

which gives the mean number of customers in the queueing system at time t. Now if we substitute these three equations into one another we have:

$$\bar{n}_t = \frac{S(t)}{t} = \frac{\bar{\tau}_t A(t)}{t} = \lambda_t \bar{\tau}_t \tag{2.88}$$

If we then go to the limit $t \to \infty$:

$$\bar{n} = \lim_{t \to \infty} \bar{n}_t \tag{2.89}$$

$$\lambda = \lim_{t \to \infty} \lambda_t$$

$$\bar{\tau} = \lim_{t \to \infty} \bar{\tau}_t$$

Then:

$$\boxed{\bar{n} = \lambda \bar{\tau} \qquad (2.90)}$$

In this derivation we have made no specific assumptions about the arrival process, the service distribution, the number of servers or the queueing discipline within the queueing system. Thus Little's Law holds even for a G/G/m queueing system.

Example: Suppose that we wish to calculate the mean waiting time in an M/M/1 queueing system. We know that

$$\bar{n} = \frac{\rho}{(1-\rho)} \tag{2.91}$$

so using Little's Law we have:

$$\boxed{\bar{\tau} = \frac{1/\mu}{(1-\rho)} \qquad (2.92)}$$

∇

Little's Law also holds if we consider only the queue and not the server. Then

$$\bar{n}_Q = \lambda \bar{\tau}_Q \tag{2.93}$$

where $\bar{n}_Q$ and $\bar{\tau}_Q$ are the analogous quantities for the waiting queue.

Little's Law existed informally long before J.D.C. Little formalized it in [LITT]. Following Little's notation, it is often written as $L = \lambda W$. Other works on this subject are mentioned at the end of this chapter in To Look Further.

2.4 Reversibility and Burke's Theorem

2.4.1 Introduction

The input to the M/M/1 queueing system is a Poisson process. But what can we say of its output? According to Burke's theorem, which was published in 1956, that output is also Poisson. This result is not immediately intuitive, despite its appealing symmetry. After all, consider the inter-departure times of the M/M/1 queueing system. As long as the queueing system is non-empty, and because we have an exponential server, these are exponentially distributed with mean duration $1/\mu$. But the problem is that sometimes the queueing system *is* empty. Then the time interval between successive departures from the queueing system is the sum of the time it takes for a customer to arrive at the empty queueing system (exponential with mean duration $1/\lambda$) and the time for that customer to be serviced (exponential with mean $1/\mu$). Thus the inter-departure times of the output process alternates between these two types of intervals. Despite

this structure, it turns out that the output process is indeed Poisson with rate λ.

To prove Burke's theorem simply, we will introduce a new concept called *reversibility*. Reversibility for a stochastic process means that when the direction of time is reversed, that is, if time flows backwards, the statistics of the process are the same as in the time normal case. As J. Walrand puts it [WALR 88]: "The technique of time reversal has proved to be a valuable tool for gaining insight into the behavior of networks. Among the successful applications of that method are simpler proofs and new examples of product-form results, as well as new results on sojourn times in networks".

We will first introduce and prove a theorem whose purpose is to allow us to establish that the number of customers in an M/M/1 queueing system as a function of time, n(t), is a reversible process. This will be used to prove Burke's theorem.

In this discussion of reversibility and Burke's theorem we will be largely following the treatment in [KELL]. This is an excellent source for more information on these concepts.

2.4.2 Reversibility

Let us make the following definition:

Definition 2.4.1: A stochastic process, X(t), is reversible if the samples $(X(t_1),X(t_2),X(t_3)\cdots X(t_m))$ has the same distribution as $(X(\tau-t_1),X(\tau-t_2)X(\tau-t_3)\cdots X(\tau-t_m))$ for every real τ (we are thinking of continuous processes) and for every $t_1,t_2,\cdots t_m$.

This definition formally states that the time reversed samples of the process X(t) are the same as the time normal samples for a reversible process.

We will now relate the reversibility of a process to the satisfication of a set of detailed balance equations. Formally we define the transition rate from state i to state j as

$$q_{ij} = \lim_{\tau\to 0}\frac{P(X(t+\tau)=j \mid X(t)=i)}{\tau} \tag{2.94}$$

for $i\neq j$ and $q_{jj}=0$.

Equilibrium will be naturally achieved if

$$\sum_{j\epsilon S} p_i\, q_{ij} = \sum_{j\epsilon S} p_j\, q_{ji} \tag{2.95}$$

or

$$p_i \sum_{j \epsilon S} q_{ij} = \sum_{j \epsilon S} p_j q_{ji} \qquad (2.96)$$

where p_j is the equilibrium probability of the jth state and S is the set of states. What is being described above is a form of global balance. What is involved in the next theorem is a form of balancing:

Theorem 2.4.1: A stationary Markov chain is reversible if and only if there is a collection of positive numbers $p_i, i \epsilon S$, which sum to one and satisfy the detailed balance equations

$$p_i q_{ij} = p_j q_{ji} \qquad (2.97)$$

for i,j ϵ S. These p_i are naturally the equilibrium state probabilities.

Proof: If the process is reversible then:

$$P(X(t)=i, X(t+\tau)=j) = P(X(t)=j, X(t+\tau)=i) \qquad (2.98)$$

But this can be rewritten as

$$P(X(t)=i)P(X(t+\tau)=j \mid X(t)=i) = P(X(t)=j)P(X(t+\tau)=i \mid X(t)=j) \qquad (2.99)$$

or:

$$p_i P(X(t+\tau)=j \mid X(t)=i) = p_j P(X(t+\tau)=i \mid X(t)=j) \qquad (2.100)$$

Dividing both sides by τ results in

$$p_i q_{ij} = p_j q_{ji} \qquad (2.101)$$

which is the desired result.

Proving the converse is a little more involved. We start by assuming that the p_i exist which satisfy the detailed balance equations. These have

to

be the equilibrium probabilities since if we sum

$$p_i q_{ij} = p_j q_{ji} \tag{2.102}$$

over j we have

$$\sum_{j \epsilon S} p_i q_{ij} = \sum_{j \epsilon S} p_j q_{ji} \tag{2.103}$$

which is just a form of global balance equation.

We will look now at the interval t ϵ [-T,T]. Suppose that the process is in state i_1, at time -T and stays in this state for a period k_1 and then jumps to state i_2 which it stays in for a period k_2 and so on, until the state is i_m for a period k_m until time T.

We will now develop the tools we will need so that we can determine the probability density of this situation. The probability density of the random variable k_1 is

$$q_{i_1} e^{-q_{i_1} k_1} \tag{2.104}$$

where:

$$q_i = \sum_{j \epsilon S} q_{ij} \tag{2.105}$$

Also, the probability that i_2 is the system state following i_1 is:

$$\frac{q_{i_1 i_2}}{q_{i_1}} \tag{2.106}$$

Similar expressions can be developed for the probability density of k_2 and the probability that i_3 follows i_2 and so forth. Finally, the probability that the state is i_m for an interval of at least k_m is

$$\int_{k_m}^{\infty} q_{i_m} e^{-q_{i_m} t} dt = -e^{-q_{i_m} t} \Big|_{k_m}^{\infty} \tag{2.107}$$

or:

$$e^{-q_{i_m} k_m} \tag{2.108}$$

We can now write the probability density of this situation:

$$p_{i_1} e^{-q_{i_1} k_1} q_{i_1 i_2} e^{-q_{i_2} k_2} \cdots q_{i_{m-1} i_m} e^{-q_{i_m} k_m} \tag{2.109}$$

This is a probability density in the variables $k_1, k_2, \cdots k_m$. One can calculate a probability of some region in these variables if one integrates over the region. Naturally, $\sum k_i = 2T$.

The last equation we need we obtain by expanding the detailed balance equation

$$p_i q_{ij} = p_j q_{ji} \tag{2.110}$$

into:

$$p_{i_1} q_{i_1 i_2} q_{i_2 i_3} \cdots q_{i_{m-1} i_m} = p_{i_m} q_{i_m i_{m-1}} q_{i_{m-1} i_{m-2}} \cdots q_{i_2 i_1}$$

This last expression tells us that the previous probability density for the situation on [-T,T] is the same as the probability density that the process starts at -T in state i_m, remains there for a period k_m and then jumps to state i_{m-1}. This continues until the process is in state i_1 for a period k_1 until T.

We now know that:

1. X(t) and X(-t) are statistically the same.

2. This means that $(X(t_1), X(t_2), X(t_3) \cdots X(t_m))$ and $(X(-t_1), X(-t_2), X(-t_3) \cdots X(-t_m))$ have the same distributions.

3. Since X(t) is stationary $(X(-t_1), X(-t_2), X(-t_3) \cdots X(-t_m))$ has the same distribution as $(X(\tau - t_1), X(\tau - t_2), X(\tau - t_3) \cdots X(\tau - t_m))$.

Thus the process is reversible and the proof is complete.

∇

The detailed balance condition says that for a reversible process states must be connected by transitions in either direction, if they are connected at all. How restrictive is this characterization? While a great many

queueing systems would not fit this characterization, some of the most important - including the M/M/1 queueing system - do.

2.4.3 Burke's Theorem

To see that the number of customers in the M/M/1 queueing system, n(t), is a reversible process, we need only to recall the existence of the local balance equation which will satisfy the condition of Theorem 2.4.1:

$$\lambda p_{n-1} = \mu p_n \tag{2.111}$$

This was previously justified by equating the probability flux flowing across a vertical boundary through the state transition diagram. In fact this same argument can be extended to show that any stationary Markov process whose state transition diagram is a tree is reversible [KELL].

This brings us to Burke's theorem [BURK]. It says that:

Theorem 2.4.2: The departure process from an M/M/1 queueing system, in equilibrium, is a Poisson process.

Proof: A realization of n(t), the number of customers in the M/M/1 queueing system, is shown in Figure 2.14. The points at which the process n(t) jumps upwards corresponds to the (Poisson) arrival process of rate λ. Since n(t) is reversible the points at which the process n(-t) jumps upwards must also represent a Poisson process of rate λ. But these points also correspond to the departure events of the queueing system for n(t). Thus the output process is Poisson with rate λ.

∇

Note how simple the concept of reversibility makes the proof. This sort of argument will work with any queueing system with a Poisson arrival process and for which n(t) is a birth-death process.

2.5 The State Dependent M/M/1 Queueing System

2.5.1 The General Solution

In all that has been said so far in this chapter the arrival rate and the service rate have been constants. Specifically, they have been independent of the number of customers in the queueing system. Many useful queueing systems can be developed if the arrival rate and/or service rate is a

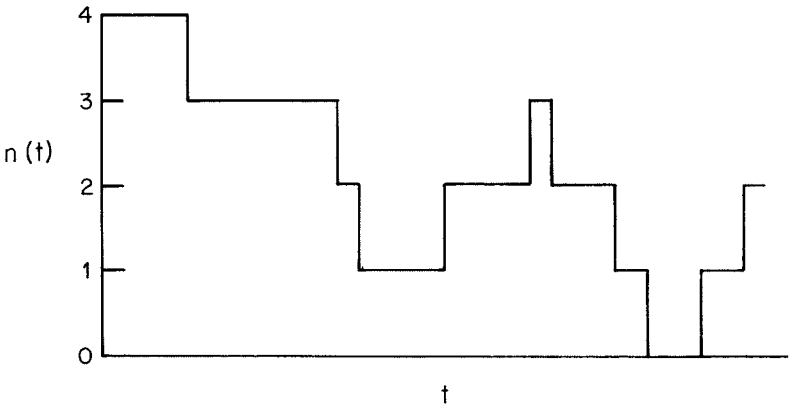

Fig. 2.14: Proving Burke's Theorem

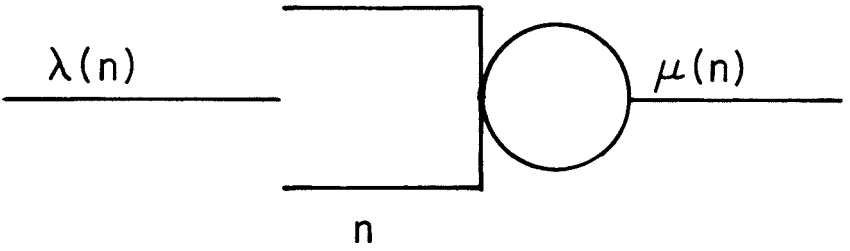

Fig. 2.15: State Dependent Queueing System

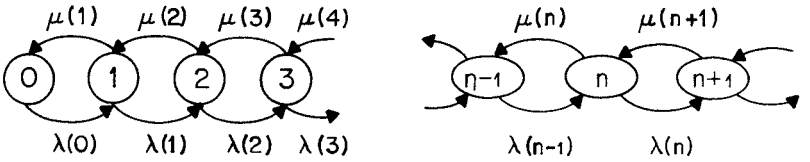

Fig. 2.16: State Dependent State Transition Diagram

function of, or dependent on, the number of customers in the queueing system. This is illustrated in Figure 2.15. The state transition diagram appears in Figure 2.16.

By drawing vertical boundaries between each pair of adjacent states and equating the flow of probability flux from left to right with the flow of probability flux from right to left yields a set of local balance equations:

$$\lambda(0)p_0 = \mu(1)p_1 \tag{2.112}$$

$$\lambda(1)p_1 = \mu(2)p_2$$

$$\lambda(2)p_2 = \mu(3)p_3$$

$$\lambda(n-1)p_{n-1} = \mu(n)p_n \tag{2.113}$$

From these we can write the recursions:

$$p_1 = \frac{\lambda(0)}{\mu(1)}p_0 \tag{2.114}$$

$$p_2 = \frac{\lambda(1)}{\mu(2)}p_1$$

$$p_3 = \frac{\lambda(2)}{\mu(3)}p_2$$

$$p_n = \frac{\lambda(n-1)}{\mu(n)}p_{n-1} \tag{2.115}$$

Substituting these recursions into each other leads to the equation:

$$\boxed{p_n = \left(\prod_{i=1}^{n} \frac{\lambda(i-1)}{\mu(i)}\right) p_0} \tag{2.116}$$

To calculate p_0 we make use of the fact that

$$\sum_{n=0}^{\infty} p_n = 1 \tag{2.117}$$

so that we can write out:

$$p_0 + \frac{\lambda(0)}{\mu(1)}p_0 + \frac{\lambda(0)\lambda(1)}{\mu(1)\mu(2)}p_0 + \frac{\lambda(0)\lambda(1)\lambda(2)}{\mu(1)\mu(2)\mu(3)}p_0 + \cdots = 1 \tag{2.118}$$

Solving for p_0 we have:

$$p_0 = \frac{1}{1 + \sum_{n=1}^{\infty} \prod_{i=1}^{n} \frac{\lambda(i-1)}{\mu(i)}} \tag{2.119}$$

2.5.2 Performance Measures

The previous two boxed equations gives us a way of calculating the equilibrium state probabilities. But why are these so important? Mainly because many practical performance measures are functions of these state probabilities. For instance, the mean *throughput* of the queueing system, or the mean rate at which customers pass through it, is:

$$\overline{\Upsilon} = \sum_{n=1}^{\infty} \mu(n)p_n \tag{2.120}$$

Note that the index starts at 1 since when n=0 the queueing system is empty and there is no contribution to throughput. The mean throughput is a weighted average of the service rates where the state probabilities serve as the weights. A second performance measure is the mean number of customers in the queueing system or:

$$\bar{n} = \sum_{n=1}^{\infty} np_n \qquad (2.121)$$

Again, this is a weighted average, of the number of customers in the queueing system with the state probabilities serving as weights.

Since for an infinite buffer M/M/1 queueing system the mean throughput is equal to the arrival rate, λ (recall Burke's theorem), we can use Little's Law to write an expression for the mean *time delay*:

$$\tau = \frac{\bar{n}}{\Upsilon} = \frac{\sum_{n=1}^{\infty} np_n}{\sum_{n=1}^{\infty} \mu(n)p_n} \qquad (2.122)$$

As a check, the numerator is dimensionless and the denominator has units of inverse seconds so τ has units of seconds.

Finally a very simple performance measure is utilization, or the probability that the queueing system is non-empty and the server is busy:

$$U = 1 - p_0 \qquad (2.123)$$

Recall from section 2.2.9 that for the infinite buffer M/M/1 queueing system with state independent service and arrival rates that $U = \lambda/\mu$.

In the next several sections we will look at specific instances of state dependent queueing systems. These include some queueing systems where, by setting certain transition rates to zero, we wind up with a finite number of states. An earlier, somewhat more extensive, collection of such queueing systems is in [KLEI 75].

2.6 The M/M/1/N Queueing System: The Finite Buffer Case

In this scenario we have a queueing system that has a limit, N, on the number of customers that can be in the system. If there are N customers in the queueing system, an arriving customer is said to be "turned away" or "lost" or "blocked". This means that the arriving customer really is lost and does not return at a later time. If such blocked customers were to return later, the arrival process would no longer be Poisson. The state dependent arrival and service rates are now:

$$\lambda(n) = \lambda \qquad n=0,1,2 \cdots N-1 \tag{2.124}$$

$$\mu(n) = \mu \qquad n=1,2,3 \cdots N$$

The state transition diagram, which has a finite number of states, is shown in Figure 2.17. Substituting the transition rates into the last section's expressions for the state probabilities yields:

$$p_n = \left(\frac{\lambda}{\mu}\right)^n p_0 \qquad 0 \leq n \leq N \tag{2.125}$$

$$p_0 = \frac{1}{\sum_{n=0}^{N} \left(\frac{\lambda}{\mu}\right)^n} \tag{2.126}$$

This expression for p_0 can also be written as:

$$p_0 = \frac{1 - \frac{\lambda}{\mu}}{1 - \left(\frac{\lambda}{\mu}\right)^{N+1}} \tag{2.127}$$

The numerator of the expression for p_0 is the same as the expression for p_0 for the M/M/1 queueing system with infinite buffer. The denominator accounts for the fact that the state space is now truncated to a finite number of states.

Therefore:

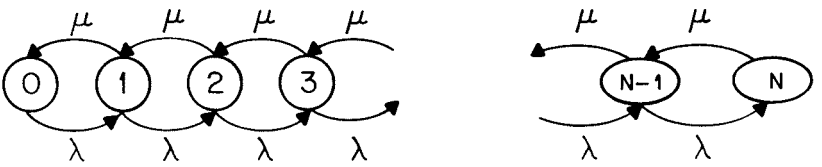

Fig. 2.17 Finite Buffer State Transition Diagram

$$p_n = \frac{1 - \frac{\lambda}{\mu}}{1 - \left(\frac{\lambda}{\mu}\right)^{N+1}} \left(\frac{\lambda}{\mu}\right)^n \qquad 0 \leq n \leq N \qquad (2.128)$$

Example: The blocking probability, p_N is the probability that the queueing system is full. The mean number of customers turned away per second is then λp_N. The following table shows the blocking probability for various values of offered load, λ/μ, when N=5:

Table 2.3: Blocking Probability	
λ/μ	Block. Prob.
0.10	9×10^{-6}
0.50	.016
0.75	.072
1.00	.166
2.00	.508
5.00	.800

As λ/μ increases, so does the blocking probability. Here the value at $\lambda/\mu = 1.00$ has to be calculated using L'Hopital's rule. For the M/M/1 queueing system with infinite buffer, λ can never be greater than μ so λ/μ is never greater than one since this would lead to an ever increasing waiting line. With a finite buffer λ can be greater than μ since excess customers are simply turned away.

As a continuation of this exercise let us tabulate the blocking probability when λ/μ=.9 and the buffer size is varied:

Table 2.4: Blocking Probability	
N	Block. Prob.
1	.474
2	.299
3	.212
5	.126
10	.051
20	.014
100	2.7×10^{-6}

As N increases, the blocking probability decreases.

∇

It can be seen that the analysis of a single queueing system with blocking is quite tractable. However it turns out that the analysis of networks of blocking queueing systems is quite difficult. The basic problem is that the presence of blocking introduces dependencies between the queueing systems that are not easily taken into account. That is, when one queueing system is full it blocks departures from the queueing system(s) which feed into it. The networks of Markovian queueing systems for which nice analytic (Chapter 3) and computational algorithms (Chapter 4) exist are ones in which there is ample buffer space. Networks of queueing systems with blocking represent an active research area [PERR 84,89].

2.7 The M/M/∞ Queueing System: Infinite Number of Servers

In this queueing system every arriving customer is assigned to its own server of rate μ. Therefore if there are n customers, the aggregate output rate is $n\mu$. The queueing system is illustrated in Figure 2.18 and the state transition diagram appears in Figure 2.19. The state dependent arrival and service rates are:

$$\lambda_n = \lambda \qquad n=0,1,2,3,\cdots \tag{2.129}$$

$$\mu_n = n\mu \qquad n=1,2,3,4,\cdots$$

Clearly, the equilibrium state probabilities are then

$$p_n = \left(\prod_{i=1}^{n} \frac{\lambda}{i\mu}\right) p_0 \tag{2.130}$$

or:

$$p_n = \frac{1}{n!}\left(\frac{\lambda}{\mu}\right)^n p_0 \tag{2.131}$$

From this it follows that:

$$p_0 = \frac{1}{1 + \sum_{i=1}^{\infty} \frac{1}{i!}\left(\frac{\lambda}{\mu}\right)^i} \tag{2.132}$$

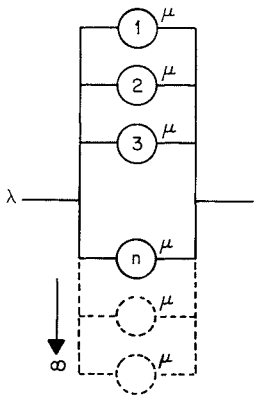

Fig. 2.18 M/M/∞ Queueing System

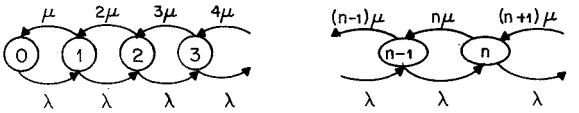

Fig. 2.19: M/M/∞ State Transition Diagram

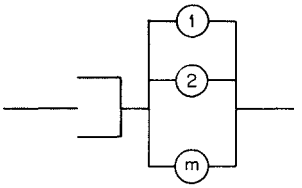

Fig. 2.20: M/M/m Queueing System

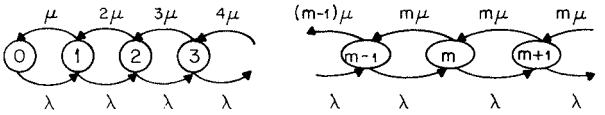

Fig. 2.21: M/M/m State Transition Diagram

This can be simplified further. We recall the familiar series expansion:

$$\sum_{i=0}^{\infty} \frac{1}{i!} x^i = e^x \tag{2.133}$$

Therefore:

$$p_0 = e^{-\lambda/\mu} \tag{2.134}$$

Thus the equilibrium state probabilities are given by:

$$\boxed{p_n = \frac{1}{n!}\left(\frac{\lambda}{\mu}\right)^n e^{-\lambda/\mu}} \tag{2.135}$$

This is just a Poisson distribution with mean $\bar{n} = \lambda/\mu$. For this model $0<\lambda/\mu<\infty$. It is the unlimited supply of servers that allows λ/μ to exceed one.

2.8 The M/M/m Queueing System: m Parallel Servers with a Queue

This queueing system is illustrated in Figure 2.20. There are m servers in parallel and a queue for additional customers. The state transition diagram appears in Figure 2.21. The state dependent transition rates are:

$$\lambda_n = \lambda \qquad n=0,1,2,3\cdots \tag{2.136}$$

$$\mu_n = \begin{cases} n\mu & 0 \le n \le m \\ m\mu & n \ge m \end{cases}$$

We can solve for the equilibrium state probabilities for $n \le m$ and $n \ge m$ separately. For $n \le m$, that is the number of customers is less than or equal to the number of servers, we have

$$p_n = \left(\prod_{i=1}^{n} \frac{\lambda}{i\mu} \right) p_0 \tag{2.137}$$

or:

$$\boxed{p_n = \frac{1}{n!} \left(\frac{\lambda}{\mu} \right)^n p_0 \qquad n \leq m \qquad (2.138)}$$

This is the same result as for the M/M/∞ system as there is one server for each customer. For n≥m, or the number of customers is greater than or equal to the number of servers, we have

$$p_n = \left(\frac{1}{m!} \left(\frac{\lambda}{\mu} \right)^m \right) \left(\prod_{i=m+1}^{n} \frac{\lambda}{m\mu} \right) p_0 \tag{2.139}$$

or:

$$\boxed{p_n = \frac{1}{m!\, m^{n-m}} \left(\frac{\lambda}{\mu} \right)^n p_0 \qquad n \geq m \qquad (2.140)}$$

Summing up over all the states we have:

$$p_0 = \left[1 + \sum_{n=1}^{m-1} \frac{1}{n!} \left(\frac{\lambda}{\mu} \right)^n + \sum_{n=m}^{\infty} \frac{1}{m!\, m^{n-m}} \left(\frac{\lambda}{\mu} \right)^n \right]^{-1} \tag{2.141}$$

or

$$p_0 = \left[1 + \sum_{n=1}^{m-1} \frac{1}{n!} \left(\frac{\lambda}{\mu} \right)^n + \frac{1}{m!} \left(\frac{\lambda}{\mu} \right)^m \sum_{n=m}^{\infty} \frac{1}{m^{n-m}} \left(\frac{\lambda}{\mu} \right)^{n-m} \right]^{-1} \tag{2.142}$$

or with a change of variables

$$p_0 = \left[1 + \sum_{n=1}^{m-1} \frac{1}{n!} \left(\frac{\lambda}{\mu} \right)^n + \frac{1}{m!} \left(\frac{\lambda}{\mu} \right)^m \sum_{n=0}^{\infty} \frac{1}{m^n} \left(\frac{\lambda}{\mu} \right)^n \right]^{-1} \tag{2.143}$$

or:

$$p_0 = \left[1 + \sum_{n=1}^{m-1} \frac{1}{n!}\left(\frac{\lambda}{\mu}\right)^n + \frac{1}{m!}\left(\frac{\lambda}{\mu}\right)^m \left(\frac{1}{1-\rho}\right)\right]^{-1} \tag{2.144}$$

In the above $\rho = \lambda/m\mu$.

In telephone engineering one quantity of interest is the probability that all the servers in this system are busy. In this context customers are actually calls. This quantity is found from what is known as Erlang's C formula (or in Europe, Erlang's formula of the second kind). We have:

$$P[queueing] = \sum_{n=m}^{\infty} p_n \tag{2.145}$$

or

$$P[queueing] = \left(\sum_{n=m}^{\infty} \frac{1}{m!m^{n-m}}\left(\frac{\lambda}{\mu}\right)^n\right)p_0 \tag{2.146}$$

Using the same technique as before this is:

$$P[queueing] = \frac{1}{m!}\left(\frac{\lambda}{\mu}\right)^m \left(\frac{1}{1-\rho}\right)p_0 \tag{2.147}$$

Substituting for p_0 this becomes:

$$P[queueing] = \frac{\frac{1}{m!}\left(\frac{\lambda}{\mu}\right)^m \left(\frac{1}{1-\rho}\right)}{\left[1 + \sum_{n=1}^{m-1} \frac{1}{n!}\left(\frac{\lambda}{\mu}\right)^n + \frac{1}{m!}\left(\frac{\lambda}{\mu}\right)^m \left(\frac{1}{1-\rho}\right)\right]} \tag{2.148}$$

2.9 The M/M/m/m Queue: A Loss System

Suppose that we take the M/M/m queueing system of the previous section and do not allow customers/calls to wait. That is, if a customer arrives and finds all the servers busy, it is lost. As Figure 2.22 shows, the queueing system simply consists of m parallel servers. The state transition diagram is like the one for the M/M/m queueing system except that it is truncated at the mth state (Figure 2.23). Therefore the state dependent transition rates are:

$$\lambda_n = \lambda \qquad n=0,1,2 \cdots m-1 \tag{2.149}$$

$$\mu_n = n\mu \qquad n=1,2,3 \cdots m$$

We have

$$p_n = \left(\prod_{i=1}^{n} \frac{\lambda}{i\mu}\right) p_0 \tag{2.150}$$

or:

$$\boxed{p_n = \frac{1}{n!}\left(\frac{\lambda}{\mu}\right)^n p_0 \qquad n \leq m} \tag{2.151}$$

Summing over all the m+1 states leads to:

$$\boxed{p_0 = \frac{1}{1 + \sum_{n=1}^{m} \frac{1}{n!}\left(\frac{\lambda}{\mu}\right)^n}} \tag{2.152}$$

A quantity of great interest in telephone engineering is p_m, the probability that all m servers are busy. For instance, the mean number of calls turned away per second is then λp_m. The expression for p_m is known as Erlang's B formula (or Erlang's formula of the first kind, in Europe) and is clearly:

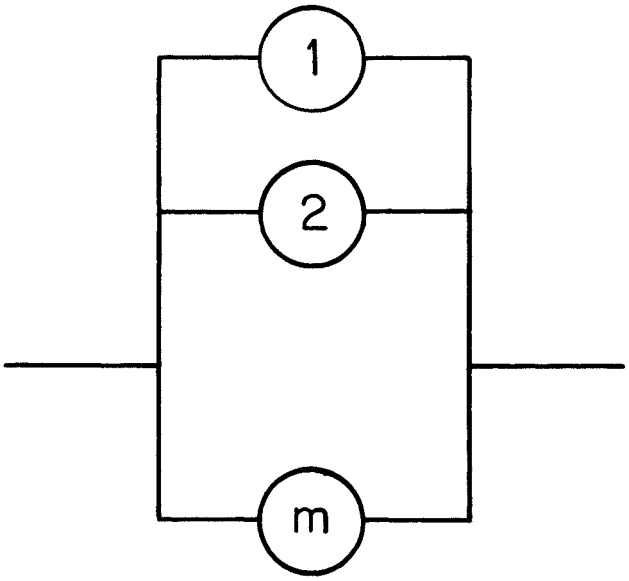

Fig. 2.22: M/M/m/m Queueing System

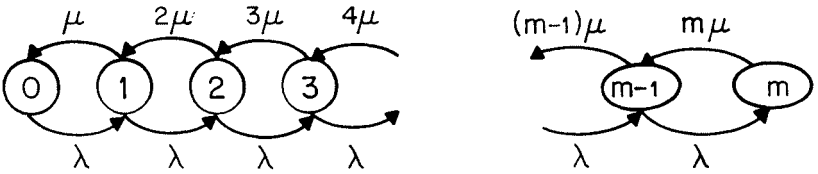

Fig. 2.23: M/M/m/m State Transition Diagram

$$p_m = \frac{\frac{1}{m!}\left(\frac{\lambda}{\mu}\right)^m}{1 + \sum_{n=1}^{m} \frac{1}{n!}\left(\frac{\lambda}{\mu}\right)^n} \qquad (2.153)$$

2.10 Central Server CPU Model

This model involves a group of m terminals which access a central CPU (Central Processing Unit). Control, in the queueing form of a customer, rotates between each terminal and the CPU. The terminals are modeled as m exponential servers, each with service rate λ. The CPU is modeled as a queueing system with a single server of rate μ. This is illustrated in Figure 2.24.

A single customer will move from its unique terminal server to the CPU queue and back to the same terminal server. The time spent by the customer at the terminal server represents the time a person spends thinking and typing at the terminal between receiving the prompt and hitting the carriage return. The time spent by the customer at the CPU is, to the user, the time between hitting the carriage return and the return of the prompt.

Even though there are really two queueing systems involved, the CPU and the terminals, the state transition diagram is still one dimensional because we have a closed queueing network. That is, the state variable can be the number of customers in the CPU, n, and then the number of active terminals is a simple function of this, m-n.

A word is necessary concerning the state description. In using the number of customers in the CPU as the state variable, we are not really specifying which customer leaves the queueing system at a departure instant, so long as some customer leaves. Thus the service discipline may be First In First Out (FIFO) or, for instance, Processor Sharing (PS). In Processor Sharing each of the n customers in the CPU receives μ/n of the service effort. This models how a CPU works on a little bit of each "job" in turn. The state transition diagram, which appears in Figure 2.25, does not really expose the fine structure of the service discipline.

To calculate the state probabilities of this system we have:

$$\lambda_n = \lambda(m-n) \qquad 0 \leq n \leq m \qquad (2.154)$$

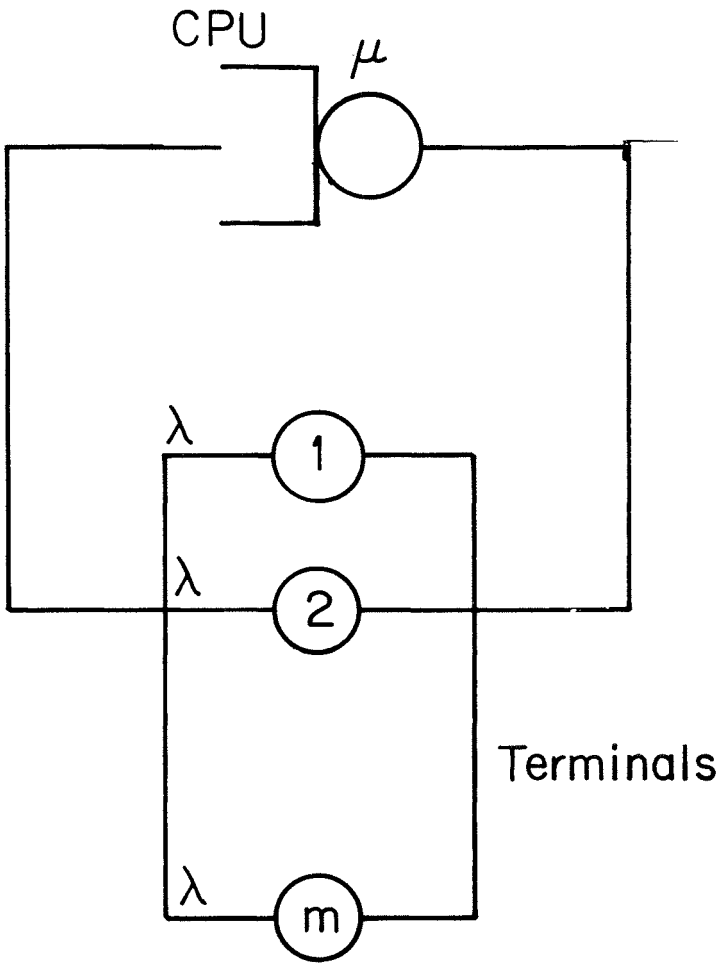

Fig. 2.24: Central Server Queueing Model

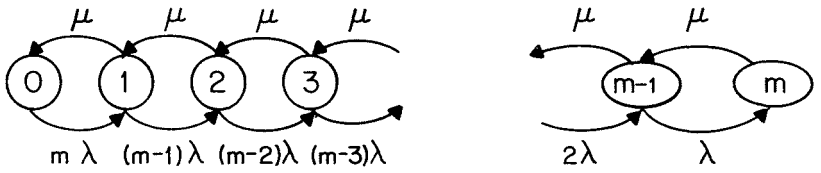

Fig. 2.25: Central Server State Transition Diagram

The state probabilities are then:

$$p_n = \left(\prod_{i=1}^{n} \frac{\lambda(m-i+1)}{\mu} \right) p_0 \tag{2.155}$$

A little thought will show that this can be rewritten as:

$$p_n = \left(\left(\frac{\lambda}{\mu} \right)^n \frac{m!}{(m-n)!} \right) p_0 \tag{2.156}$$

$$p_0 = \frac{1}{\left[\sum_{n=0}^{m} \left(\frac{\lambda}{\mu} \right)^n \frac{m!}{(m-n)!} \right]}$$

Here 0! is defined as 1.

2.11 Transient Solution of the M/M/1/∞ Queueing System

2.11.1 The Technique

The differential equations for the M/M/1/∞ queueing system are:

$$\frac{dP_n(t)}{dt} = -(\lambda+\mu)P_n(t) + \lambda P_{n-1}(t) + \mu P_{n+1}(t) \tag{2.157}$$

$$\frac{dP_0(t)}{dt} = -\lambda P_0(t) + \mu P_1(t) \tag{2.158}$$

These equations are easy to solve in the equilibrium case as the derivatives are then equal to zero and a set of equations which are linear in the state probabilities results. In this section we will look at the more complicated transient case. Specifically, we will look at the case when there are initially i customers in the queueing system. The question is then what is the behavior of the queueing system as time proceeds.

The solution technique we will use is one that can be applied to a variety of queueing systems. It makes use of the moment generating function:

$$P(z,t) = \sum_{n=0}^{\infty} P_n(t) z^n \tag{2.159}$$

Basically, the state probabilities form a sequence of numbers that can be transformed, like any numerical sequence, into a "frequency" like domain. It turns out to be relatively easy to obtain the frequency domain solution. This solution is then inverted to provide the solution to the state probabilities.

The details of the solution technique are:

A. Take the nth equation and multiply both sides by z^n. Then sum the equations up to n= ∞.

B. Take the resulting single equation and identify the moment generating function throughout the equation. It may be necessary to add and subtract a finite number of terms to explicitly obtain the moment generating function in places.

C. Use any "boundary" equation(s) to eliminate unknowns (i.e. state probabilities).

D. Solve for the moment generating function. If a differential equation results, repeat the analogs of steps A-D using Laplace transforms.

E. Invert the moment generating function, or if this is not possible, calculate moments.

We will now make this methodology a little more real in terms of the M/M/1/∞ transient analysis problem. We mention in passing that the approach for this problem first appeared in [BAIL]. Our treatment will be similar to that appearing in [GROS] and [KLEI 75].

2.11.2 The Solution

We first take the differential equation of the previous section, multiply both sides by z^n and sum from n=1 to ∞:

$$\sum_{n=1}^{\infty} \frac{dP_n(t)}{dt} z^n = -\lambda \sum_{n=1}^{\infty} P_n(t) z^n - \mu \sum_{n=1}^{\infty} P_n(t) z^n \tag{2.160}$$

$$+\lambda \sum_{n=1}^{\infty} P_{n-1}(t)z^n + \mu \sum_{n=1}^{\infty} P_{n+1}(t)z^n$$

Recall that to get proper moment generating functions the indexes must start at n=0 so we have:

$$\sum_{n=0}^{\infty} \frac{dP_n(t)}{dt} z^n - \frac{dP_0(t)}{dt} = \tag{2.161}$$
$$-\lambda \sum_{n=0}^{\infty} P_n(t)z^n + \lambda P_0(t) - \mu \sum_{n=0}^{\infty} P_n(t)z^n + \mu P_0(t)$$
$$+\lambda \sum_{n=1}^{\infty} P_{n-1}(t)z^n + \mu \sum_{n=0}^{\infty} P_{n+1}(t)z^n - \mu P_1(t)$$

But three of the terms here are just the boundary differential equation at n=0 and so drop out, leaving:

$$\sum_{n=0}^{\infty} \frac{dP_n(t)}{dt} z^n = \tag{2.162}$$
$$-\lambda \sum_{n=0}^{\infty} P_n(t)z^n - \mu \sum_{n=0}^{\infty} P_n(t)z^n + \mu P_0(t)$$
$$+\lambda \sum_{n=1}^{\infty} P_{n-1}(t)z^n + \mu \sum_{n=0}^{\infty} P_{n+1}(t)z^n$$

We can rewrite this further:

$$\sum_{n=0}^{\infty} \frac{dP_n(t)}{dt} z^n = \tag{2.163}$$
$$-\lambda \sum_{n=0}^{\infty} P_n(t)z^n - \mu \sum_{n=0}^{\infty} P_n(t)z^n + \mu P_0(t)$$
$$+\lambda z \sum_{n=1}^{\infty} P_{n-1}(t)z^{n-1} + \mu z^{-1} \sum_{n=0}^{\infty} P_{n+1}(t)z^{n+1}$$

Now identifying the moment generating functions (which sometimes requires adding and subtracting terms) we have:

$$\frac{\partial P(z,t)}{\partial t} = \tag{2.164}$$
$$-\lambda P(z,t) - \mu(P(z,t) - P_0(t))$$

$$+ \lambda z P(z,t) + \mu z^{-1}(P(z,t) - P_0(t))$$

This can be simplified:

$$\frac{\partial P(z,t)}{\partial t} = \tag{2.165}$$
$$(1 - z)(\mu z^{-1} - \lambda)P(z,t) + (1 - z)(-\mu z^{-1})P_0(t)$$

$$\boxed{\frac{\partial P(z,t)}{\partial t} = (1 - z)z^{-1}\Big((\mu - \lambda z)P(z,t) - \mu P_0(t)\Big)} \tag{2.166}$$

The initial condition to this equation involves i customers at time 0:

$$P_n(0+) = \begin{cases} 1 & n = i \\ 0 & n \neq i \end{cases} \tag{2.167}$$

or:

$$P(z,0+) = \sum_{n=0}^{\infty} P_n(0+)z^n = z^i \tag{2.168}$$

To solve this partial differential equation, we will use Laplace transforms which we define as:

$$L\{P_i(t)\} = P_i^*(s) = \int_0^{\infty} P_i(t)e^{-st}\,dt \tag{2.169}$$

$$L\{P(z,t)\} = P^*(z,s) = \int_0^{\infty} P(z,t)e^{-st}\,dt$$

For the left hand side of the boxed equation we will integrate by parts to find its transform:

$$\int u\,dv = uv - \int v\,du \tag{2.170}$$

Specifically:

$$\int_0^\infty e^{-st}\frac{\partial P(z,t)}{\partial t}dt = \left(e^{-st}P(z,t)\right)\Big|_0^\infty + \int_0^\infty se^{-st}P(z,t)dt \qquad (2.171)$$

$$= -P(z,0) + sP^*(z,s) \qquad (2.172)$$

$$= -z^i + sP^*(z,s) \qquad (2.173)$$

Now we will take the Laplace transform of the boxed differential equation and multiply out the constants so that we have:

$$-z^{i+1} + szP^*(z,s) = \qquad (2.174)$$
$$(\mu - \lambda z - \mu z + \lambda z^2)P^*(z,s) - \mu(1-z)P_0{}^*(s)$$

We can now solve for $P^*(z,s)$:

$$\boxed{P^*(z,s) = \frac{z^{i+1} - \mu(1-z)P_0{}^*(s)}{-\lambda z^2 + (s+\lambda+\mu)z - \mu} \qquad (2.175)}$$

At this point we must determine $P_0{}^*(s)$ and invert the transform. The interested reader can see [GROS] for the details. The well known result for the time evolution of the state probabilities is

$$\boxed{\begin{aligned} &P_n(t) = \\ &e^{-(\lambda+\mu)t}\Big\{\rho^{(n-i)/2}I_{n-i}(2t\sqrt{\lambda\mu}) + \rho^{(n-i-1)/2}I_{n+i+1}(2t\sqrt{\lambda\mu}) \\ &(1-\rho)\rho^n \sum_{l=n+i+2}^{\infty} \rho^{-l/2}I_l(2t\sqrt{\lambda\mu})\Big\} \end{aligned} \qquad (2.176)}$$

where

$$\rho = \lambda/\mu$$

$$n(0) = i \text{ customers}$$

and $I_l(\ )$ is the modified Bessel function of the first kind of order l:

$$I_l(x) = \sum_{k=0}^{\infty} \frac{\left(\frac{x}{2}\right)^{l+2k}}{(l+k)!k!} \tag{2.177}$$

2.11.3 Speeding Up The Computation

In calculating the expressions for $P_n(t)$ in the previous box the main difficulty is in calculating the infinite summation of Bessel functions. We will now present an efficient procedure for the calculation of $P_n(t)$ which appeared in 1980 [JONE] and is due to Jones, Cavin and Johnston at Texas A&M University. The approach is to show that all but a finite number of terms in the infinite summation can be expressed in terms of the *Q functions.* This function is normally associated with the field of hypothesis testing.

If

$$Q(a,b) = \int_b^{\infty} \exp\left[-\frac{a^2+x^2}{2}\right] I_0(ax)xdx \tag{2.178}$$

Then

$$1 - Q(a,b)$$

is the circular coverage function. The function Q(a,b) is known as the Q-function [SCHW 66] in the areas of radar and communication. It is the probability that $\{(X^2+Y^2)^{1/2} > b\}$ where X and Y are independent normal (Gaussian) random variables with variances of one and means which satisfy $(\overline{X}^2+\overline{Y}^2)^{1/2} = a \geq 0$. Note that these are the same a and b that appears in the argument of Q(a,b).

The Q function was first developed by Marcum [MARC]. There are existing algorithms for its calculation [DIDO].

The key to efficient computation is to make use of the expansion:

$$Q(a,b) = \exp\left(-\frac{(a^2+b^2)}{2}\right)\sum_{m=0}^{\infty}\left(\frac{a}{b}\right)^m I_m(ab) \qquad (2.179)$$

This is done by first letting $a=\sqrt{2\mu t}$ and $b=\sqrt{2\lambda t}$ so that:

$$Q(\sqrt{2\mu t},\sqrt{2\lambda t}) = \exp(-(\lambda+\mu)t)\sum_{m=0}^{\infty}\left(\frac{\mu}{\lambda}\right)^{m/2} I_m(2t\sqrt{\lambda\mu}) \qquad (2.180)$$

The relevant infinite summation can then be set equal to a finite number of terms:

$$\exp(-(\lambda+\mu)t)\sum_{l=n+i+2}^{\infty}\rho^{-l/2}I_l(2t\sqrt{\lambda\mu}) = Q(\sqrt{2\mu t},\sqrt{2\lambda t}) - \exp(-(\lambda+\mu)t)\sum_{l=0}^{n+i+1}\rho^{-l/2}I_l(2t\sqrt{\lambda\mu}) \qquad (2.181)$$

Here $\rho=\lambda/\mu$. Substituting this into the boxed expression for $P_n(t)$ of the previous section leads to:

$$P_n(t) = (1-\rho)\rho^n\ Q(\sqrt{2\mu t},\sqrt{2\lambda t}) + \exp(-(\lambda+\mu)t)\left\{\rho^{(n-i)/2}I_{n-i}(2t\sqrt{\lambda\mu}) + \rho^{(n-i-1)/2}I_{n+i+1}(2t\sqrt{\lambda\mu}) - (1-\rho)\rho^n\sum_{l=0}^{n+i+1}\rho^{-l/2}I_l(2t\sqrt{\lambda\mu})\right\} \qquad (2.182)$$

Computational results reported in [JONE] indicate that this expression is up to seven times faster to evaluate than the original expression with the infinite summation. Some additional references dealing with computation in the transient case appear in To Look Further at the end of this chapter.

2.12 The M/G/1 Queueing System

2.12.1 Introduction

We have been able to get some very specific results throughout this chapter by assuming Poisson arrivals and an exponential server(s). This has been due to the memoryless nature of these models. Surprisingly, some very simple and powerful results can be derived for the M/G/1 queueing system where we assume that arrivals are again Poisson but that service times are described by an arbitrary ("general") probability distribution. These are the famous Pollaczek-Kinchin formulas [POLL] [KHIN]. The first one we will look at is for the mean number of customers in the queueing system. The second one we will look at is a transform equation for the state probabilities.

2.12.2 Mean Number in the Queueing System

Introduction

Our basic approach will be to write a difference equation for the number in the queueing system immediately after the ith departure. By taking the expectation of the square of this expression we will be able to develop the Pollaczek-Khinchin mean value formula. A key point is that the state probabilities at these departure instants are in fact equal to the state probabilities at any time instant. This is something that will be assumed true for now but is proved explicitly in section 2.12.3. This approach which we take was first described by David Kendall in 1951 [KEND]. However the actual mean value expression goes back to Pollaczek in 1930 [POLL] and Khinchin in 1932 [KHIN] where it was found by means of more complex derivations. As many other authors have done ([GROS],[HAMM],[KLEI 75]) we will expand on Kendall's approach.

The Recursion

Let us start. We will look at the M/G/1 queueing system at the instants during which departures occur. The number in the system *immediately after* the i+1st departure instant is n_{i+1}. This number includes both those customers in the queue and the customer in service. This number is also equal to the number in the system immediately after the ith departure instant minus 1 (because of the departure of a customer at the i+1st departure instant) plus the number that arrive into the system between the ith and i+1st departure instants. This number of arrivals we will call a_{i+1}. For this equation to work there must be at least one

customer in the system at the ith departure instant. Thus:

$$n_{i+1} = n_i - 1 + a_{i+1} \qquad n_i > 0 \tag{2.183}$$

Of course we still have to take care of the case $n_i = 0$. This is actually simpler than the case $n_i > 0$. The number in the queueing system at the i+1st departure instant is just the number of arrivals since the ith departure instant (when the system was empty). So:

$$n_{i+1} = a_{i+1} \qquad n_i = 0 \tag{2.184}$$

Bringing this together we have:

$$n_{i+1} = \begin{cases} n_i - 1 + a_{i+1} & n_i > 0 \\ a_{i+1} & n_i = 0 \end{cases} \tag{2.185}$$

We would like to express this as a single equation. To do this, we'll introduce the unit step function:

$$u(n_i) = \begin{cases} 1 & n_i > 0 \\ 0 & n_i = 0 \end{cases} \tag{2.186}$$

Now we can rewrite the equations for n_{i+1} as:

$$\boxed{n_{i+1} = n_i - u(n_i) + a_{i+1} \qquad (2.187)}$$

If we square both sides of this recursion and take expectations we have:

$$E[(n_{i+1})^2] = E[(n_i - u(n_i) + a_{i+1})^2] \tag{2.188}$$

$$E[(n_{i+1})^2] = \qquad (2.189)$$

$$E[n_i^2] + E[(u(n_i))^2] + E[a_{i+1}^2]$$

$$- 2E[n_i\, u(n_i)] + 2E[n_i\, a_{i+1}] - 2E[u(n_i) a_{i+1}]$$

In order to solve for the Pollaczek-Khinchin formula we need to solve for each of these terms, one by one. We will now do this:

Sundry Results

$E[(n_{i+1})^2]$, $E[n_i^2]$:

These two terms should be equal in equilibrium so that they will cancel in the above boxed equation.

$E[(u(n_i))^2]$:

Because of the nature of the step function $u(n_i)^2 = u(n_i)$ so we can replace $E[(u(n_i))^2]$ with $E[u(n_i)]$. But what is $E[u(n_i)]$? We will have to digress in order to solve for it. Now:

$$E[u(n_i)] = \sum_{n=0}^{\infty} u(n_i) P[n_i = n] \qquad (2.190)$$

$$E[u(n_i)] = \sum_{n=1}^{\infty} P[n_i = n] \qquad (2.191)$$

$$E[u(n_i)] = P[busy] \qquad (2.192)$$

Luckily, we can easily determine $P[busy]$ for any queueing system. An argument appearing in [KLEI 75] will be used. Consider an interval I. The server will be busy for a time period equal to:

$$I - Ip_0$$

The number of customers served is

$$(I - Ip_0)\mu$$

where μ is the mean service rate of the queueing system. In equilibrium the quantity above should be equal to the number of arrivals:

$$\lambda I = (I - Ip_0)\mu \tag{2.193}$$

Rearranging this one has

$$\boxed{\rho = 1 - p_0 \qquad (2.194)}$$

Note that we have made no restrictive assumptions about either the arrival process or the service times. Thus this result is valid for a G/G/1 queueing system, that is one with an arbitrary arrival process and service time distribution.

We can now bring all this together to say that:

$$E[(u(n_i))^2] = E[u(n_i)] = P[busy] = 1 - p_0 = \rho \tag{2.195}$$

$2E[n_i u(n_i)]$:

Clearly $n_i u(n_i) = n_i$ so we can say:

$$2E[n_i u(n_i)] = 2E[n_i] \tag{2.196}$$

$2E[u(n_i)a_{i+1}]$:

The number of customers that arrive into the system between the ith and i+1st departure instant, a_{i+1}, is independent of the number in the queueing system immediately after the ith departure instant, n_i. Thus:

$$E[u(n_i)a_{i+1}] = E[u(n_i)]E[a_{i+1}] \tag{2.197}$$

We already know that $E[u(n_i)]=\rho$. To determine $E[a_{i+1}]$ we can look at the original recursion for $E[n_{i+1}]$

$$n_{i+1} = n_i - u(n_i) + a_{i+1} \tag{2.198}$$

and take the expectation of both sides:

$$E[n_{i+1}] = E[n_i] - E[u(n_i)] + E[a_{i+1}] \tag{2.199}$$

In equilibrium $E[n_{i+1}]=E[n_i]$ so we are left with:

$$E[a_{i+1}] = E[u(n_i)] = \rho \tag{2.200}$$

We can now say that:

$$2E[u(n_i)a_{i+1}] = 2E[u(n_i)]E[a_{i+1}] = 2\rho^2 \tag{2.201}$$

$2E[n_i a_{i+1}]$:

Using the same argument as before, the two quantities are statistically independent so

$$2E[n_i a_{i+1}] = 2E[n_i]E[a_{i+1}] = 2E[n_i]\rho \tag{2.202}$$

Putting It All Together

Now we can substitute these sundry results into the equation:

$$E[(n_{i+1})^2] = \tag{2.203}$$

$$E[n_i^2] + E[(u(n_i))^2] + E[a_{i+1}^2]$$

$$- 2E[n_i u(n_i)] + 2E[n_i a_{i+1}] - 2E[u(n_i)a_{i+1}]$$

This results in:

$$0 = 0 + \rho + E[a_{i+1}{}^2] - 2E[n_i] + 2E[n_i]\rho - 2\rho^2 \qquad (2.204)$$

Solving for $E[n_i]$ we have:

$$E[n_i] = \frac{\rho + E[a_{i+1}{}^2] - 2\rho^2}{2(1-\rho)} \qquad (2.205)$$

We will assume for now, and prove in section 2.12.3, that the expected number of customers in the queueing system at the departure instants is the same as the expected number of customers at *any* instant of time:

$$E[n_i] = E[n] \qquad (2.206)$$

We can also say, because the input is Poisson, that the expected number of arrivals between two successive departure instants is a time invariant quantity:

$$E[a_{i+1}{}^2] = E[a^2] \qquad (2.207)$$

We thus have:

$$\boxed{E[n] = \frac{\rho + E[a^2] - 2\rho^2}{2(1-\rho)}} \qquad (2.208)$$

We almost have a simple expression for the mean number in the M/G/1 queueing system in equilibrium, or $E[n]$. What is lacking is a simple expression for $E[a^2]$:

What is $E[a^2]$?

The quantity "a" is the number of arrivals during a typical servicing period in equilibrium. Proceeding as in [GROS] we can write:

$$E[a^2] = VAR[a] + (E[a])^2 \qquad (2.209)$$

$$E[a^2] = VAR[a] + \rho^2 \tag{2.210}$$

We need to solve for VAR [a]. Let s be the service time. Then an interesting relationship [GROS] for two random variables X and Y that we can make use of is [PARZ]:

$$VAR[Y] = E[VAR(Y \mid X)] + VAR[E(Y \mid X)] \tag{2.211}$$

In terms of our application this becomes:

$$VAR[a] = E[VAR(a \mid s)] + VAR[E(a \mid s)] \tag{2.212}$$

We can solve for the two components of this sum:

$$E[VAR(a \mid s)] = E[\lambda s] = \lambda E[s] = \rho \tag{2.213}$$

$$VAR[E(a \mid s)] = VAR[\lambda s] = \lambda^2 {\sigma_s}^2 \tag{2.214}$$

Here ${\sigma_s}^2$ is the variance of the service time. So we have

$$VAR[a] = \rho + \lambda^2 {\sigma_s}^2 \tag{2.215}$$

and

$$\boxed{E[a^2] = \rho + \lambda^2 {\sigma_s}^2 + \rho^2} \tag{2.216}$$

The Pollaczek–Khinchin Mean Value Formula

From before we have:

$$E[n] = \frac{\rho + E[a^2] - 2\rho^2}{2(1-\rho)} \tag{2.217}$$

Substituting the preceding boxed expression for $E[a^2]$ leads to

$$E[n] = \frac{\rho + \rho + \lambda^2 {\sigma_s}^2 + \rho^2 - 2\rho^2}{2(1-\rho)} \tag{2.218}$$

or:

$$E[n] = \frac{2\rho - \rho^2 + \lambda^2 {\sigma_s}^2}{2(1-\rho)} \qquad (2.219)$$

This is one form of the Pollaczek-Khinchin mean value formula. We can get another form by expressing E[n] as

$$E[n] = \frac{2\rho - 2\rho^2 + \rho^2 + \lambda^2 {\sigma_s}^2}{2(1-\rho)} \qquad (2.220)$$

or

$$E[n] = \rho + \frac{\rho^2 + \lambda^2 {\sigma_s}^2}{2(1-\rho)} \qquad (2.221)$$

What is amazing about the Pollaczek-Khinchin mean value formula is that it expresses E[n] as a function of the mean arrival rate, mean service rate and variance of the service time distribution. One might think that since the service time distribution is general, the formula should include higher moments of the service time distribution. But this is not the case. The formula depends only on the mean and variance of the service time distribution.

We note in passing that Little's Law can be used in conjunction with this mean value formula to develop expressions for the mean waiting time, $\overline{r}$.

Example: In the M/D/1 queueing system the arrivals are Poisson and the service time is a fixed, deterministic quantity. This is actually a good model for the processing of fixed length packets in a communications network.

Let's make a comparison. For the M/M/1 queueing system the Pollaczek-Khinchin mean value formula reduces to the result we had previously:

$$E[n] = \frac{\rho}{1-\rho}$$

For the M/D/1 queueing system $\sigma_s^2=0$ so:

$$E[n] = \rho + \frac{\rho^2}{2(1-\rho)}$$

We can make a table comparing E[n] for both types of queueing systems for various values of ρ:

Table 2.5: E[n]		
ρ	M/M/1	M/D/1
.1	.111	.106
.2	.250	.225
.3	.429	.364
.4	.667	.533
.5	1.00	.750
.6	1.50	1.05
.7	2.33	1.52
.8	4.00	2.40
.9	9.00	4.95
.99	99.0	50.0

Note that the expected number in the table is smaller for the M/D/1 queueing system. In fact we can show, with some algebraic manipulation, that for the M/D/1 queueing system:

$$E[n] = \frac{\rho}{1-\rho} - \frac{\rho^2}{2(1-\rho)}$$

The E[n] is equal to that for the M/M/1 queueing system, less a positive quantity. Thus E[n] is always smaller for an M/D/1 queueing system compared to an M/M/1 queueing system.

Although the M/D/1 queueing system is a good model for packet processing, the problem is that we only have simple analytical results for a single such system. Because networks of M/D/1 queues are analytically intractable, the memoryless M/M/1 queueing system is often used as an approximation in its place.

∇

2.12.3 Why We Use Departure Instants

An essential part of the previous derivation of the Pollaczek-Khinchin mean value formula is our assumption that the state probabilities at departure instants are the same as the state probabilities at any instant. Following the method used in [GROS 85], we will show that this is true.

We will take a realization of the queueing process over a long interval. The number in the queueing system at time t will be n(t). The interval will stretch from time 0 to time T.

The number of arrivals in the interval (0,T) that occur when the number of customers in the system is n will be called $a_n(T)$. The number of departures bringing the system from state n+1 to state n in the interval (0,T) will be called $d_n(T)$.

If one thinks about the movement of the system state in the usual one dimensional state transition diagram for a birth death system one can see that:

$$| a_n(T) - d_n(T) | \leq 1 \tag{2.222}$$

It should also be true from first principles that

$$d(T) = a(T) + n(0) - n(T) \tag{2.223}$$

where d(T) and a(T) are the total number of departures and arrivals, respectively, over (0,T). We can now define the state probabilities for the departure instants as:

$$\lim_{T\to\infty} \frac{d_n(T)}{d(T)} \qquad n=1,2,3... \tag{2.224}$$

This empirical definition of the state probability becomes exact in the limit. Now:

$$\frac{d_n(T)}{d(T)} = \frac{a_n(T) + d_n(T) - a_n(T)}{a(T) + n(0) - n(T)} \tag{2.225}$$

The denominator can be recognized as the equation we just discussed, two equations previous. The numerator was created by adding and subtracting $a_n(T)$ to $d_n(T)$. As $T\to\infty$ $d_n(T)$-$a_n(T)$ will never be greater than one, as has been mentioned. Also, because we have a stationary process if n(0) is finite, so will be n(T). Thus as $T\to\infty$, $a(T)$ and $a_n(T)$ will

$\rightarrow\infty$

and:

$$\lim_{T\to\infty}\frac{d_n(T)}{d(T)} = \lim_{T\to\infty}\frac{a_n(T)}{a(T)} \tag{2.226}$$

What this proves directly is that the state probabilities seen at the departure instants are the same as the state probabilities seen at the arrival instants. But going back to our Poisson process coin flipping analogy, the arrival instants are random points in time and independent of the queueing system state. Thus it should seem reasonable to say that the state probabilities at arrival instants are the same as the state probabilities seen at any time. This completes the proof that looking at state probabilities at departure instants for the M/G/1 queueing system is the same as looking at state probabilities for this queueing system at *any* instant.

∇

In Kendall's original 1951 paper [KEND] he called the instants of departures "regeneration points". As he put it: "An epoch is a point of regeneration if a knowledge of the state of the process at that particular epoch has the characteristic Markovian consequence that a statement of the past history of the process loses all its predictive value." He went on further to say that "A Markov process, therefore, is precisely a process for which "every" epoch is a point of regeneration."

In the printed discussion that followed this paper D.V. Lindley formally expressed a regeneration point as a point, t_0, where for all $t > t_0$:

$$DIST\{X(t) \mid X(t_0)\} \equiv DIST\{X(t) \mid X(\tau) \text{ for } all\ \tau \leq t_0\}$$

There is more than one set of regeneration points. Kendall used the set of departure instants in his paper. In the subsequent discussion Lindley suggested the instants when the queueing system becomes empty. He also mentioned the use of the time instants when a customer had been in service x seconds. For simplicity of solution though, the choice of departure instants is probably best.

2.12.4 Probability Distribution of the Number in the Queueing System

What does the Markov chain imbedded at the times of departure look like? Let us define a matrix of transition probabilities as:

$$\mathbf{P} = [P_{rs}] \equiv P[n_{i+1}=s \mid n_i=r] \tag{2.227}$$

Here n_i is the number in an M/G/1 queueing system immediately after the ith departure. To get from $n_i=r$ to $n_{i+1}=s$ there must be s-r+1 arrivals between the ith and the i+1st departure instants. The "+1" in s-r+1 is necessary because the number in the queueing system is reduced by one due to the departure at the i+1st departure instant. Let us set

$$[P_{rs}] = k_{s-r+1} = k_{\#\ arrivals} \tag{2.228}$$

where "# arrivals" is the number of customers that arrive between the ith and the i+1st departure instants. We are using k's because this is the original notation appearing in Kendall's 1951 paper. What we now have is a state transition matrix that looks like this:

Table 2.6: $\mathbf{P} = [p_{rs}] =$

	0	1	2	3	4	·
0	k_0	k_1	k_2	k_3	k_4	·
1	k_0	k_1	k_2	k_3	k_4	·
2	0	k_0	k_1	k_2	k_3	·
3	0	0	k_0	k_1	k_2	·
4	0	0	0	k_0	k_1	·
·	·	·	·	·	·	·

The sum of the k's in each row must be one as this covers the probability of going from state r to *some* (and possibly the same) state. The matrix is also of infinite dimension but this will not cause any problems in what follows.

How was the placement of matrix entries chosen? In the top row one goes from 0 customers in the queueing system at the ith departure instant to 0,1,2,3... customers at the i+1st departure instant. Thus we need k_0, k_1, k_2, k_3..... arrivals. In the second row, for instance, to go from 1 customer at the ith departure instant to 3 customers at the i+1st departure instant there must be 3 arrivals during this time. That is, we start with 1 customer at the end of the ith departure instant, three more customers arrive for a total of 4 customers and one customer leaves at the end of the

i+1st departure instant for a final total of 3 customers.

We can express k_j, the probability of j arrivals between the ith and the i+1st departure instants, explicitly as

$$k_j = \int_0^\infty Prob[j \ arrivals \mid time \ s]b(s)ds \tag{2.229}$$

where b(s) is the service time density. The variable s is the time between the ith and the i+1st departure instants. This time is a constant in equilibrium. Now since the arrival stream is Poisson, we can substitute the Poisson distribution into the integral for:

$$k_j = \int_0^\infty \frac{(\lambda s)^j e^{-\lambda s}}{j!} b(s)ds \tag{2.230}$$

The basic idea is that for a known service time density, b(s), we can plug it into the above equation and perform the integration to solve for k_j. We began this section by asking what the Markov chain imbedded at the times of departures would look like. The answer is illustrated in Figure 2.26. Here are shown all the transitions leaving the nth state. The k_j associated with each transition is the probability that one leaves the state via that transition as opposed to another transition at the ith departure instant. It is *not* a transition rate, as in the previous state transition diagrams.

Let us now ask a second question. Now that we have this Markov chain, can we calculate the probability of there being j customers in the queueing system at the departure instants? The answer is yes. We will use an approach described by [GROS]. An alternate approach appears in [KLEI].

We know from the previous section that the state probabilities at these instants are in fact the same as the state probabilities at any instant. The first step is to write the equilibrium equations

$$\pi = \pi \mathbf{P} \tag{2.231}$$

where

$$\pi = [\pi_0 \ \pi_1 \ \pi_2 \ \pi_3 \ \cdots] \tag{2.232}$$

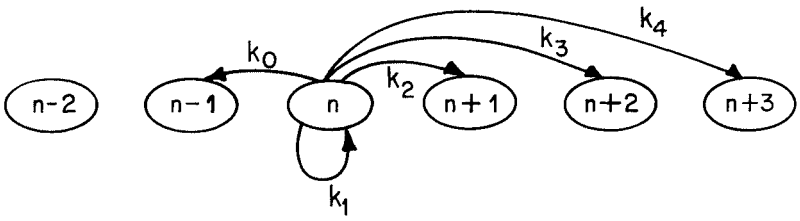

Fig. 2.26: Imbedded Markov Chain for M/G/1

are the state probabilities. We know the structure of **P**. The equation $\pi=\pi\mathbf{P}$ gives rise to equations of the form:

$$\pi_0 = \pi_0 k_0 + \pi_1 k_0 \tag{2.233}$$

$$\pi_1 = \pi_0 k_1 + \pi_1 k_1 + \pi_2 k_0$$

$$\pi_2 = \pi_0 k_2 + \pi_1 k_2 + \pi_2 k_1 + \pi_3 k_0$$

$$\pi_3 = \pi_0 k_3 + \pi_1 k_3 + \pi_2 k_2 + \pi_3 k_1 + \pi_4 k_0$$

These equations can be compactly summarized, where r and s are integers, as:

$$\pi_r = \pi_0 k_r + \sum_{s=1}^{r+1} \pi_s k_{r-s+1} \qquad r=0,1,2,3... \tag{2.234}$$

The second step is to make use of transforms. First we introduce the change of variables t=s-1 or s=t+1. This gives us:

$$\pi_r = \pi_0 k_r + \sum_{t=0}^{r} \pi_{t+1} k_{r-t} \tag{2.235}$$

We will also introduce the transform of the state probabilities

$$\Pi(z) = \sum_{i=0}^{\infty} \pi_i z^i \tag{2.236}$$

and the transform of the k_j's:

$$K(z) = \sum_{i=0}^{\infty} k_i z^i \tag{2.237}$$

The k's, after all, are a sequence of real (positive) numbers like any other sequence and so there is no reason why we can not take their transform. Finally, the summation in the above difference equation for π_r

can

be seen to be "almost" a convolution. That is, it is similar to

$$x_i = \sum_{j=0}^{\infty} f_j g_{i-j} \tag{2.238}$$

for which we know that

$$X(z) = F(z)G(z) \tag{2.239}$$

where X(z), F(z), and G(z) are the one sided (i=0 to ∞) transforms of x_i, f_i and g_i, respectively.

Let us now take the one-sided transform of the previous difference equations for π_r. This is:

$$\Pi(z) = \pi_0 K(z) + Z\{\pi_{t+1}\}K(z) \tag{2.240}$$

We can take the transform of the convolution, in spite of the finite index, because π_i and k_i are only defined for $i \geq 0$. In the preceding equation:

$$Z\{\pi_{t+1}\} = \sum_{t=0}^{\infty} \pi_{t+1} z^t = z^{-1} \sum_{t=0}^{\infty} \pi_{t+1} z^{t+1} \tag{2.241}$$

$$Z\{\pi_{t+1}\} = z^{-1} \sum_{t=-1}^{\infty} \pi_{t+1} z^{t+1} - z^{-1}\pi_0 \tag{2.242}$$

$$Z\{\pi_{t+1}\} = z^{-1}\Pi(z) - z^{-1}\pi_0 \tag{2.243}$$

So substituting this into the transform equation yields

$$\Pi(z) = \pi_0 K(z) + [z^{-1}\Pi(z) - z^{-1}\pi_0]K(z) \tag{2.244}$$

or

$$\Pi(z)(1 - z^{-1}K(z)) = \pi_0(1 - z^{-1})K(z) \tag{2.245}$$

Solving for $\Pi(z)$ gives us:

$$\Pi(z) = \frac{\pi_0(1 - z^{-1})K(z)}{1 - z^{-1}K(z)} \tag{2.246}$$

Multiplying by -z/-z results in:

$$\Pi(z) = \frac{\pi_0(1 - z)K(z)}{K(z) - z} \tag{2.247}$$

To solve for π_0 one can make use of the fact that $\Pi(1)=1$, $K(1)=1$ and $K'(1)=\lambda/\mu$ and use of L'Hopital's rule to obtain $\pi_0 = 1 - \lambda/\mu = 1 - \rho$ so:

$$\boxed{\Pi(z) = \frac{(1-\rho)\,(1-z)\,K(z)}{K(z) - z} \qquad (2.248)}$$

This is the result that we are seeking. It expresses the transform of the probabilities of the number in the M/G/1 queueing system as a function of the transform of the k_j's. Recall that we obtain the k_j's from an integral equation which includes b(s), the service time distribution.

To Look Further

An English translation of Markov's original 1907 paper on Markov models appears in an appendix of [HOWA].

Stidham's [STID] 1974 extended proof of Little's Law is discussed in [COOP]. Further work on Little's Law appears in [BRUM], [HEYM], [MIYA], [JEWE] and [EILO].

The proof of Theorem 2.4.2 using the concept of reversibility is due to E. Reich [REIC], 1957.

[GROS] has a wealth of information on transient solutions of queueing systems. [KLEI 75] also describes the transient solution for the M/M/1/∞ queueing system. Ackroyd [ACKR] describes an alternative method to make the calculation of $P_n(t)$ in the M/M/1/∞ transient case more efficient. It involves a discrete Fourier transform. A discussion of its merits relative to the approach in section 2.11.3 appears in [CHIE]. A method for the computation of the transient M/M/1 queueing system cumulative

distribution function, probability distribution function and mean, which uses generalized Q functions, is described by Cantrell in [CANT]. A method for calculating performance measures based on numerical integration appears in [ABAT]. Finally, numerical approaches to the transient analysis problem are discussed in [REIB].

More examples of the use of the moment generating function appear in chapter 12 of [SCHW 87] and in chapter 4 of [KLEI 75].

Problems

2.1 Use the coin flipping analogy of section 2.1.1 to show intuitively that random splits of a Poisson process and a joining of independent Poisson processes are Poisson.

2.2 Arrive at a set of differential equations describing the time evolution of the Poisson process from

$$P_n(t+\Delta t) = P_n(t)(1-\lambda\Delta t) + P_{n-1}(t)(\lambda\Delta t)$$

$$P_0(t+\Delta t) = P_0(t)(1 - \lambda\Delta t)$$

in a step by step manner.

2.3 a) Show that the solution of

$$\frac{dP_1(t)}{dt} = -\lambda P_1(t) + \lambda e^{-\lambda t}$$

for the Poisson process is:

$$P_1(t) = \lambda t e^{-\lambda t}$$

b) Show that the solution of

$$\frac{dP_2(t)}{dt} = -\lambda P_2(t) + \lambda^2 t e^{-\lambda t}$$

for the Poisson process is:

$$P_2(t) = \frac{\lambda^2 t^2}{2} e^{-\lambda t}$$

2.4 Arrive at a set of differential equations describing the M/M/1 queueing system from

$$P_n(t+\Delta t) =$$

$$P_n(t)(1-\lambda\Delta t)(1-\mu\Delta t)+P_{n-1}(t)(\lambda\Delta t)+P_{n+1}(t)(\mu\Delta t)$$

$$P_0(t+\Delta t) = P_0(t)(1-\lambda\Delta t) + P_1(t)(\mu\Delta t)$$

in a step by step manner.

2.5 Derive the differential equations describing the evolution of probability of the system state for an M/M/1 queueing system when the service rate, μ_n, is a function of the number in the queue, n, and the arrival rate, λ_n, is also a function of the number in the queue [KLEI 75].

2.6 Find the equation for an M/M/1 queueing system for $dP_0(t)/dt$ by examining the flows across a boundary about state 0.

2.7 Verify Table 2.2.

2.8 Prove that for a stationary Markov process the probability flux across a cut across a state transition diagram balances [KELL].

2.9 Using the result of problem 2.8, prove that if the state transition diagram of a stationary process is a tree, then the process is reversible [KELL].

2.10 Show that for an M/M/1 queueing system in equilibrium the number of customers in the queue at t_0, $n(t_0)$, is independent of the departure process prior to t_0 [REIC] [KELL].

2.11 For the M/M/1/N queueing system use L'Hopital's rule to calculate the blocking probability when N=5 and $\lambda/\mu = 1$.

2.12 For the state independent M/M/1 queueing system find the ratio p_{100}/p_0 when λ=1.0 and μ=2.0. Comment on potential numerical problems.

2.13 Consider a state independent M/M/1 queueing system with ρ=.1, .5, .9 Calculate the utilization, average number in the queueing system, the variance of the number in the queueing system and two standard deviations from the average of the number in the queueing system.

2.14 Compute the $P[queueing]$ using the Erlang C formula for the values in the following table when m=3:

λ/μ	$\lambda/m\mu$	$P[queueing]$
.5	.166	
1.0	.333	
1.5	.500	
2.0	.666	
2.5	.833	
2.9	.966	

2.15 Compute p_m, the probability that all servers are busy, for an M/M/m/m queueing loss system for various values of λ/μ. Use Erlang's B formula to fill in the following table when m=3:

λ/μ	p_m
.5	
1.0	
2.0	
3.0	
6.0	

2.16 In finding the probability distribution of the number of customers in the M/G/1 queueing system one arrives at:

$$\Pi(z) = \frac{\pi_0(1-z)K(z)}{K(z)-z}$$

Solve for π_0 using L'Hopital's rule and $\pi(1)=1$, $K(1)=1$ and $K'(1)=\lambda/\mu$.

2.17 **Computer Project:** Write a computer program to generate data for plots of the Erlang B and Erlang C formula.

Erlang B: Plot the probability that all servers are busy versus ρ where $\rho=\lambda/\mu$. Do this for m=1,2,3. The vertical scale can be a linear scale. Let ρ vary from 0 to 2m on the horizontal scale.

Erlang C: Plot P[queueing] versus ρ where $\rho=\lambda/m\mu$. Plot curves for m=1,2,3. The vertical scale for P[queueing] should be a log scale. Let ρ vary from 0 to 1 on the horizontal scale.

Chapter 3: Networks of Queues

3.1 Introduction

In this chapter we will consider networks of queues. Simple analytical results are usually only possible for Markovian queueing networks. We will start by establishing the product form solution for the equilibrium state probabilities for such networks in section 3.2. The existence of the product form solution basically means that the joint state probability can be expressed as a simple product of functions associated with a network's individual queues. In the case of open queueing networks of state independent queues these functions are simply the marginal state probabilities so that it seems that the queues act *as if* they were independent. This interesting observation was first noted by J.R. Jackson [JACK 57] in the original product form paper for open networks in 1957 and later generalized in [JACK 64]. Later, in 1967, W.J. Gordon and G.F. Newell [GORD] demonstrated the existence of the product form solution for closed networks. In 1975 F. Baskett, K.M. Chandy, R.R. Muntz and F.G. Palacios [BASK 75] generalized the families of queueing networks known to have the product form solution.

We will also look at the intriguing phenomena of local balance. There is always a "local balancing" of flows on pairs of state transition diagram transitions when the product form solution exists.

In section 3.3 a relatively new interpretation of the product form solution will be presented. We will see that the pattern of flow of probability flux on the state transition diagram has a very definite, decomposable structure. In finding the product form solution for a state we are really solving localized building blocks within the diagram in a recursion from a reference state to the state of interest.

Many networks of practical interest do not have a product form solution. However the equilibrium probabilities of quite a few such non-product form models can be determined recursively, or at least in a decomposed fashion. Two such classes of models are presented in section 3.4.

This chapter concludes with three case studies involving distributed queueing networks. Problems of this type represent a new challenge for the application of queueing theory. In section 3.5 we present an interesting case study based on recent work by M. Karol, M. Hluchyj (now at Codex) and S. Morgan of AT&T Bell Laboratories. It involves the calculation of throughput for a space division packet switch. It is a good case study as it

brings together ideas concerning networks of queues, non-Markovian queues and transform techniques in the context of a problem of practical interest.

In section 3.6 a second case study is presented based on work by Y.C. Jenq [JENQ], formerly of AT&T Bell Laboratories and now at Tektronix. It involves a Banyan interconnection network with single buffers at each switching element. It is an interesting case study as it shows one way to approach a distributed queueing system with blocking.

Finally, in section 3.7, a case study based on recent work by M. Garrett of Bell Communications Research and S.-Q. Li of the University of Texas at Austin is presented. It deals with the placement of erasure nodes in the DQDB metropolitan area network. An ingenious continuous formulation of the problem allows the calculation of optimal erasure station location and system throughput.

3.2 The Product Form Solution

3.2.1 Introduction

We will look at both the case of open and closed networks. In each case we will set up a generalized global balance equation for the typical state. Then it will be shown that this balance equation is satisfied by the "product form" solution. The approach taken here appears in [KOBA] and [SCHW 87].

3.2.2 Open Networks

3.2.2.1 The Global Balance Equation

We will assume that we have a network of M queues with a single external source for customers and a single external destination for customers. The routing through this network is random. The network is illustrated in Fig. 3.1. In this figure the probability of a customer leaving queue i for queue j is r_{ij}. The probability of a customer leaving the source for queue i is r_{si}. The probability of a customer leaving queue i for the destination is r_{id}.

The source produces customers in a Poisson manner with constant rate λ. The service times at the ith queue are independent exponential random variables with rate μ_i.

Now for a little bit about the state description. Let $\underline{1}_i$ be the vector

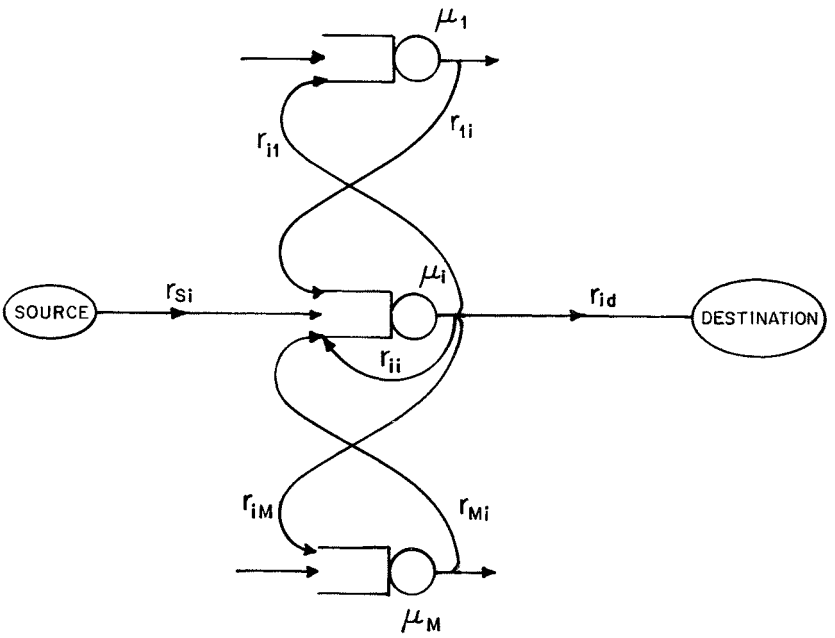

Fig. 3.1: Open Queueing Network

$$\underline{1}_i = (0,0,0, \cdots 0,1,0, \cdots 0) \tag{3.1}$$

with a 1 in the ith position. The state description is the vector:

$$\underline{n} = (n_1, n_2, n_3, \cdots n_i, \cdots n_M) \tag{3.2}$$

Let the vector $\underline{n}+\underline{1}_i$ be $\underline{n}$ with an additional customer in the ith position. Let $\underline{n}-\underline{1}_i$ be $\underline{n}$ with one customer removed from the ith position. Let $\underline{n}+\underline{1}_j-\underline{1}_i$ be $\underline{n}$ with one customer added to queue j and one customer removed from queue i.

If one equates the net flow out of state $\underline{n}$ with the net flow into state $\underline{n}$ one has [SCHW 87] the global balance equation at the state $\underline{n}$:

$$\left(\lambda + \sum_{i=1}^{M} \mu_i\right) p(\underline{n}) = \sum_{i=1}^{M} \lambda r_{si}\, p(\underline{n}-\underline{1}_i) \tag{3.3}$$
$$+ \sum_{i=1}^{M} \mu_i r_{id}\, p(\underline{n}+\underline{1}_i) + \sum_{i=1}^{M}\sum_{j=1}^{M} \mu_j r_{ji}\, p(\underline{n}+\underline{1}_j-\underline{1}_i)$$

The left hand side of this equation corresponds to leaving state $\underline{n}$ through arrivals or departures for the queues. The three terms on the right hand side cover all the ways in which one may enter the state: through an arrival, a departure or a transfer between two queues.

In order to suggest a solution to this equation, we will make a digression:

3.2.2.2 The Traffic Equations

Let the average throughput through queue i be θ_i. Then in equilibrium it must be true for each queue that:

$$\theta_i = r_{si}\lambda + \sum_{j=1}^{M} r_{ji}\theta_j \qquad i=1,2,3...M \tag{3.4}$$

This set of simultaneous linear equations are known as the "traffic equations". Their unique solution is the average throughput for each

queue. The equation above says that the average throughput through queue i is equal to the average arrival flow from the source plus the flows from all the other queues including queue i.

Example: Consider the open queueing network of Fig. 3.2. The routing probabilities are shown on the diagram. It is left as an exercise to the reader to show that:

$$\theta_1 = 2\lambda \qquad \theta_2 = \frac{8}{3}\lambda$$

Each of the queue's throughputs are greater than λ as the feedback paths allow customers to move through each queue several times before leaving the network.

∇

3.2.2.3 The Product Form Solution

To solve the previous global balance equation we will re-arrange the traffic equations as

$$\lambda r_{si} = \theta_i - \sum_{j=1}^{M} r_{ji}\theta_j \qquad i=1,2,3...M \tag{3.5}$$

and use it to eliminate λr_{si} in the global balance equation [CHEN]. Substituting, one has:

$$\left(\lambda + \sum_{i=1}^{M} \mu_i\right) p(\underline{n}) = \sum_{i=1}^{M} \theta_i p(\underline{n}-\underline{1}_i) - \sum_{i=1}^{M}\sum_{j=1}^{M} r_{ji}\theta_j p(\underline{n}-\underline{1}_i)$$

$$+ \sum_{i=1}^{M} \mu_i r_{id} p(\underline{n}+\underline{1}_i) + \sum_{i=1}^{M}\sum_{j=1}^{M} \mu_j r_{ji} p(\underline{n}+\underline{1}_j-\underline{1}_i) \tag{3.6}$$

We will now propose using the identity $\theta_i p(\underline{n}-\underline{1}_i)=\mu_i p(\underline{n})$ in this equation. This is really a local balance equation for the ith queue. It is a multidimensional generalization of the local balance equations used in the previous chapter for the M/M/1 queueing system. It is equivalent with $\theta_j p(\underline{n}-\underline{1}_i) = \mu_j p(\underline{n}-\underline{1}_i+\underline{1}_j)$. Substituting these equations:

$$\left(\lambda + \sum_{i=1}^{M} \mu_i\right) p(\underline{n}) = \sum_{i=1}^{M} \mu_i p(\underline{n}) - \sum_{i=1}^{M}\sum_{j=1}^{M} r_{ji}\mu_j p(\underline{n} - \underline{1}_i+\underline{1}_j)$$

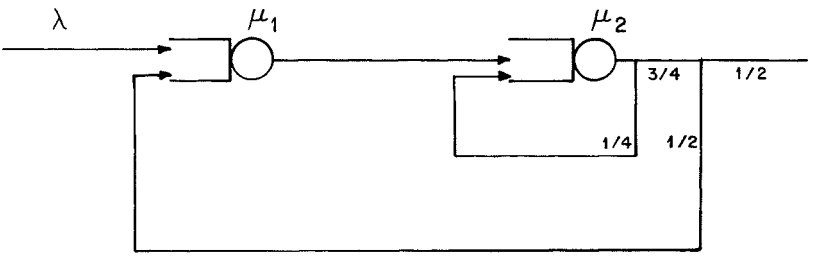

Fig. 3.2: Example Open Queueing Network

$$+ \sum_{i=1}^{M} \mu_i r_{id} p(\underline{n}+\underline{1}_i) + \sum_{i=1}^{M} \sum_{j=1}^{M} \mu_j r_{ji} p(\underline{n}+\underline{1}_j-\underline{1}_i) \tag{3.7}$$

Canceling terms results in:

$$\lambda p(\underline{n}) = \sum_{i=1}^{M} \mu_i r_{id} p(\underline{n}+\underline{1}_i) \tag{3.8}$$

Using a third, equivalent, local balance equation $\mu_i p(n+\underline{1}_i) = \theta_i p(\underline{n})$ results in

$$\lambda p(\underline{n}) = \sum_{i=1}^{M} r_{id} \theta_i p(\underline{n}) \tag{3.9}$$

or

$$\lambda = \sum_{i=1}^{M} r_{id} \theta_i \tag{3.10}$$

This equation, which relates the net flow into the network with the net flow out of the network is obviously true. Thus what has been demonstrated is that the local balance equation $\theta_i p(\underline{n}-\underline{1}_i) = \mu_i p(\underline{n})$ satisfies the previous global balance equation. To get a closed form solution we can rearrange the local balance equation as:

$$p(\underline{n}) = \frac{\theta_i}{\mu_i} p(\underline{n}-\underline{1}_i) \tag{3.11}$$

$$p(\underline{n}) = \frac{\theta_i}{\mu_i} p(n_1, n_2, n_3, \cdots n_i-1, \cdots n_M) \tag{3.12}$$

Continuing the recursion to zero out the ith term results in:

$$p(\underline{n}) = \left(\frac{\theta_i}{\mu_i}\right)^{n_i} p(n_1, n_2, n_3, \cdots 0, \cdots n_M) \tag{3.13}$$

If one does this for each of the queues one has:

$$p(\underline{n}) = p(\underline{0}) \prod_{i=1}^{M} \left(\frac{\theta_i}{\mu_i} \right)^{n_i} \qquad (3.14)$$

To solve for p(0) we can use the normalization of probability [SCHW 87]:

$$\sum_{n} p(\underline{n}) = p(\underline{0}) S = 1 \qquad (3.15)$$

$$S = \sum_{n} \left(\prod_{i=1}^{M} \left(\frac{\theta_i}{\mu_i} \right)^{n_i} \right) \qquad (3.16)$$

Interchanging the summation and product:

$$S = \prod_{i=1}^{M} \sum_{n_i=0}^{\infty} \left(\frac{\theta_i}{\mu_i} \right)^{n_i} \qquad (3.17)$$

$$S = \prod_{i=1}^{M} \left(1 - \frac{\theta_i}{\mu_i} \right)^{-1} \qquad (3.18)$$

So the *marginal* probability that a queue is empty is:

$$p_i(0) = \left(1 - \frac{\theta_i}{\mu_i} \right) \qquad (3.19)$$

Note the similarity to equation (2.53). And:

$$p(\underline{n}) = \prod_{i=1}^{M} p_i(n_i) \qquad (3.20)$$

$$p_i(n_i) = \left(1 - \frac{\theta_i}{\mu_i} \right) \left(\frac{\theta_i}{\mu_i} \right)^{n_i}$$

This is the "product form solution" for the equilibrium state probabilities first observed by Jackson. Naturally one must first solve the traffic equations for the θ_i's in order to compute $p(\underline{n})$. One special case worth mentioning is a number of queues in series (tandem) fed by a Poisson source of rate λ. Then $\theta_i = \lambda$ for i=1,2,3...M.

It should also be mentioned that the product form solution is still valid if there are multiple sources and destinations.

Example: Let's calculate the product form solution for the network of Fig. 3.2. From the previous example it is known that $\theta_1 = 2\lambda$ and $\theta_2 = \frac{8}{3}\lambda$. Thus

$$p(\underline{n}) = p(\underline{0})\left(\frac{2\lambda}{\mu_1}\right)^{n_1}\left(\frac{(8/3)\lambda}{\mu_2}\right)^{n_2}$$

or:

$$p(\underline{n}) = \left(1-\frac{2\lambda}{\mu_1}\right)\left(\frac{2\lambda}{\mu_1}\right)^{n_1}\left(1-\frac{(8/3)\lambda}{\mu_2}\right)\left(\frac{(8/3)\lambda}{\mu_2}\right)^{n_2}$$

∇

What has always attracted attention about the product form solution is that its similarity to the decomposition of a joint probability in terms of its marginal probabilities for independent random variables suggests that, in a product form network, queues act "independently" of one another. Actually what it really means is that if you took a "snapshot" of the network at a time instant the number of customers found in each queue are independent of one another. However if you took snapshots of the network at two, close, time instants you would find strong correlations between the numbers of customers in the queues at the two time instants [DISN].

3.2.3 Local Balance

We just succeeded in solving the global balance equation for state $\underline{n}$ of an open queueing network by using the local balance equation $\theta_i p(\underline{n}-\underline{1}_i) = \mu_i p(\underline{n})$. Local balance is a phenomena peculiar to queueing networks. Suppose, for simplicity of argument, that we have a series (tandem) network so that $\theta_i = \lambda$. Then $\lambda p(\underline{n}-\underline{1}_i) = \mu_i p(\underline{n})$. What this says is that the amount of probability flux flowing on pairs of transitions incident to a state are equal. This is a stronger form of balancing than global balance where we simply equate the flow into a state totaled over *all* the

incoming transitions with the flow out of the state totaled over *all* the outgoing transitions. With local balance the more specific action is being taken of equating flows on individual pairs of transitions.

Global balance has an analog in resistive circuit theory in the form of Kirchoff's current conservation law. Local balance has no such analog. Intuitively, this is because transition values, unlike resistor values, are labeled in a patterned manner that gives rise to local balance.

If one has a state with several transitions, how does one know which ones are "paired" for local balance? The simple rule to follow is [SAUE 81]:

> The flow of probability flux into a state due to an arrival to a queue equals the flow out of the same state due to a departure from the same queue.

While we have used the case of series queues, what has been said holds generally for product form networks. In section 3.3 local balance will be explained in an integrated fashion.

3.2.4 Closed Queueing Networks

Consider a closed queueing network of M queues with a finite population of N customers. Let μ_i be the service rate of the ith exponential queue. Routing is, again, random with r_{ij} being the probability of a customer departing queue i entering queue j. We have the same state

$$\underline{n} = (n_1, n_2, n_3, \cdots n_i, \cdots n_M) \tag{3.21}$$

as for the open network and the same use of the unit vector:

$$\underline{1}_i = (0,0,0, \cdots 1, \cdots 0) \tag{3.22}$$

If one equates the net flow out of state $\underline{n}$ with the net flow into state $\underline{n}$ one has [KOBA] the global balance equation at state $\underline{n}$:

$$\sum_{i=1}^{M} \mu_i p(\underline{n}) = \sum_{i=1}^{M} \sum_{j=1}^{M} \mu_j r_{ji} p(\underline{n}+\underline{1}_j-\underline{1}_i) \tag{3.23}$$

The left hand side of this equation corresponds to customers leaving state $\underline{n}$ through queue departures while the right hand side corresponds to entering state $\underline{n}$ from state $\underline{n}+\underline{1}_j-\underline{1}_i$ through a transfer of customers from queue j to queue i.

The traffic equations take the form:

$$\theta_i = \sum_{j=1}^{M} r_{ji}\theta_j \qquad i=1,2,3...M \qquad (3.24)$$

Interestingly, these traffic solutions do not have a unique solution. Any solution $(\theta_1,\theta_2,\theta_3,\cdots\theta_M)$ multiplied by a real number k, $(k\theta_1,k\theta_2,k\theta_3,\cdots k\theta_M)$ is also a solution.

To solve the global balance equation rewrite the traffic equations as:

$$1 = \sum_{j=1}^{M} r_{ji}\frac{\theta_j}{\theta_i} \qquad i=1,2,3...M \qquad (3.25)$$

Now multiply μ_i in the global balance equation by this identity for one:

$$\sum_{i=1}^{M}\sum_{j=1}^{M}\mu_i r_{ji}\frac{\theta_j}{\theta_i}p(\underline{n}) = \sum_{i=1}^{M}\sum_{j=1}^{M}\mu_j r_{ji}\,p(\underline{n}+\underline{1}_j-\underline{1}_i) \qquad (3.26)$$

This can be rewritten as:

$$\sum_{i=1}^{M}\sum_{j=1}^{M} r_{ji}\left\{\frac{\theta_j}{\theta_i}\mu_i p(\underline{n}) - \mu_j p(\underline{n}+\underline{1}_j-\underline{1}_i)\right\} = 0 \qquad (3.27)$$

Clearly this equation will be satisfied if the following local balance equation holds:

$$\frac{\theta_j}{\theta_i}\mu_i p(\underline{n}) = \mu_j p(\underline{n}+\underline{1}_j-\underline{1}_i) \qquad i,j=1,2,3...M \qquad (3.28)$$

Rearranging one has:

$$p(\underline{n}) = \left(\frac{\theta_i}{\mu_i}\right)\left(\frac{\theta_j}{\mu_j}\right)^{-1} p(\underline{n}+\underline{1}_j-\underline{1}_i) \qquad i,j=1,2,3...M \tag{3.29}$$

This is equivalent to:

$$p(\underline{n}) = \left(\frac{\theta_i}{\mu_i}\right) p(\underline{n}-\underline{1}_i) \tag{3.30}$$

Expanding:

$$p(\underline{n}) = \left(\frac{\theta_i}{\mu_i}\right) p(n_1,n_2,n_3, \cdots n_i-1, \cdots n_M) \tag{3.31}$$

Continuing the recursion to zero out the ith term one obtains:

$$p(\underline{n}) = \left(\frac{\theta_i}{\mu_i}\right)^{n_i} p(n_1,n_2,n_3, \cdots 0, \cdots n_M) \tag{3.32}$$

Doing this for each queue yields:

$$p(\underline{n}) = \prod_{i=1}^{M}\left(\frac{\theta_i}{\mu_i}\right)^{n_i} p(\underline{0}) \tag{3.33}$$

We will use the normalization of probability to find an explicit expression for $p(\underline{0})$:

$$\sum_{\underline{n}} p(\underline{n}) = p(\underline{0})G(N) = 1 \tag{3.34}$$

$$p(\underline{0}) = \frac{1}{G(N)} = \frac{1}{\sum_{\underline{n}}\left(\prod_{i=1}^{M}\left(\frac{\theta_i}{\mu_i}\right)^{n_i}\right)} \tag{3.35}$$

Here G(N) is a "normalization constant" that is computed over the finite state space of the closed queueing network. Because the summation for G(N) does not go over an infinite number of states, as it does open queueing networks, it does *not* factor into a product $p_1(0)p_2(0)p_3(0) \cdots p_M(0)$.

We thus have the product form solution for closed queueing networks:

$$p(\underline{n}) = \frac{1}{G(N)} \prod_{i=1}^{M} \left(\frac{\theta_i}{\mu_i} \right)^{n_i} \qquad (3.36)$$

Though not explicitly stated, G(N) is also a function of M. As discussed in chapter four, the efficient computation of G(N) is important in the convolution algorithm for calculating closed queueing network performance measures.

Finally, it should be mentioned that even though the θ_i's do not have a unique solution, as long as the same values are used for the θ_i's consistently in (3.36) for both G(N) and the term in brackets then $p(\underline{n})$ will be correct.

3.2.5 The BCMP Generalization

In 1975 F. Baskett, K.M. Chandy, R.R. Muntz, F.G. Palacios published a generalization of the product form solution in terms of networks consisting of servers with four possible service disciplines. The four service disciplines that lead to the product form solution are [BASK 75]:

Service discipline 1: The service discipline is First In First Out (FIFO). All customers, regardless of class, have the same exponential service time distribution. The service rate can be a function of the *total number of customers in the individual queue.*

Service Discipline 2: The service discipline is processor sharing. That is, each of n customers receives 1/n of the service effort. Each class of customer may have a distinct service time distribution (with a rational Laplace transform).

Service Discipline 3: In this service discipline there is one server for each customer. Each class of customer may have a distinct service time distribution (with a rational Laplace transform).

Service Discipline 4: The service discipline is preemptive-resume Last In First Out (LIFO). Each class of customer may have a distinct service time distribution (with a rational Laplace transform).

The state description of these product form networks can be quite detailed (jump ahead to Fig. 3.16 for an example), state dependent arrivals are allowed, there are different classes of customers and some classes may be closed and some may be open. The product form solution for such a

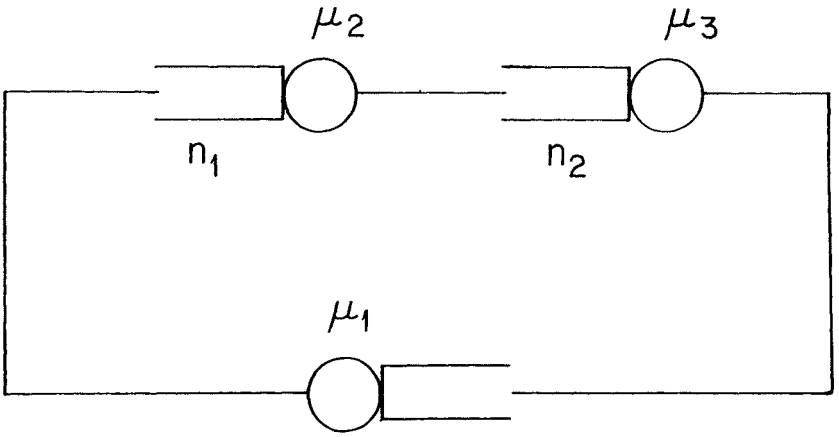

Fig. 3.3: Cyclic Queueing Network

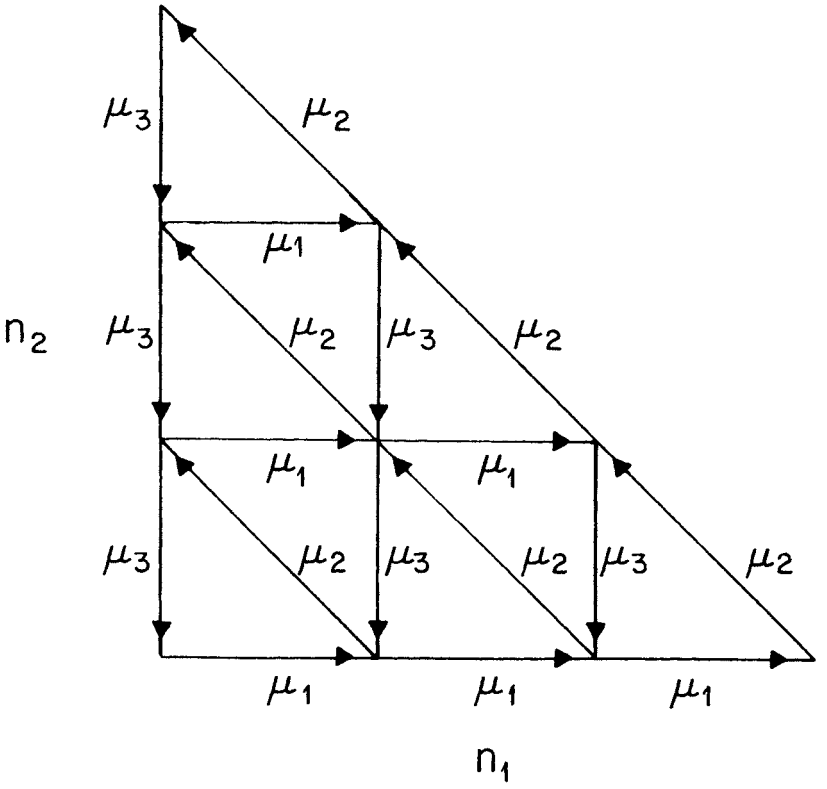

Fig. 3.4: State Transition Diagram of Fig. 3.3

complex model appears in [BASK 75].

3.3 Algebraic Topological Interpretation of P.F. Solution

3.3.1 Introduction

Why does the product form solution exist and how can one explain its simple form? During the mid 1980's an interesting and aesthetically pleasing interpretation of the existence of the product form solution became possible. The basic idea is the realization that the state transition diagrams of product form networks can be decomposed into an aggregation of elementary "building blocks". Because these building blocks do not interact with each other in a substantial way, they can be solved in isolation for the relative state probabilities. The product form solution, then, for a specific equilibrium state probability in terms of a reference probability is really a recursion that patches together these building block solutions along a "path" from the state of interest back to the reference state.

This algebraic topological interpretation of the product form solution is important for several reasons. It gives a constructive means for arriving at the product form solution. It helps to explain why certain models have a product form solution and why others do not. It becomes possible to modify or select models so that they have a product form solution [HAMI 86a,86b,89] or to construct product form models which provide performance bounds for the more difficult to analyze non-product form networks [VAND].

In what follows these ideas will be expanded upon. The sources for this material are from the work of A. Lazar of Columbia University [LAZA 84a, 84b, 84c], I. Wang of the New Jersey Institute of Technology [WANG 86] and the author.

3.3.2 A First Look at Building Blocks

Consider the cyclic three queue network in Figure 3.3. The service times are exponential random variables. The state transition diagram is illustrated in Figure 3.4. The state description for this model is two dimensional since this is a closed network. That is, if n_1 and n_2 are the numbers of customers in the upper queues, then $N-n_1-n_2$ is automatically the number in the lower queue where N is the total number of customers in the network.

Now let us look at the local balance equations of this product form network. Recalling the rule that allows us to pair transitions for local

balance, in Figure 3.5 the paired transitions which exhibit local balance are shown. Observing the lower left corner of the state transition diagram one can see that the local balance pairings mean that the flow from (0,0) to (1,0) equals that from (1,0) to (0,1). But the flow from (1,0) to (0,1) must equal that from (0,1) to (0,0). Finally the flow from (0,1) to (0,0) must equal that from (0,0) to (1,0).

Thus an alternative view of local balance is that there is a circular flow of probability flux along the edges of the triangular sub-element: (0,0) to (1,0) to (0,1) and back to (0,0). With this circular flow, each edge has the same magnitude of probability flux.

Continuing with this logic, each triangular sub-element (or building block) in the state transition diagram has a circular flow about its edge. This is illustrated in Figure 3.6.

Because of the decomposition of the flow of probability flux that is thus possible, one can effectively remove one building block at a time from the remaining building blocks without disturbing its flow pattern, except for a renormalization. The renormalization is necessary so that the probabilities of the remaining states sum to one. A completely decomposed state transition diagram is shown in Figure 3.7.

If one wants to solve for p_{21} for instance, one solves building block C in this last figure to obtain the local balance equation

$$\mu_1 p_{11} = \mu_2 p_{21} \tag{3.37}$$

and the solution:

$$p_{21} = \frac{\mu_1}{\mu_2} p_{11} \tag{3.38}$$

Solving building block E one obtains the local balance equation

$$\mu_3 p_{11} = \mu_1 p_{10} \tag{3.39}$$

and the solution:

$$p_{11} = \frac{\mu_1}{\mu_3} p_{10} \tag{3.40}$$

Finally, the building block D yields the local balance equation

$$\mu_1 p_{00} = \mu_2 p_{10} \tag{3.41}$$

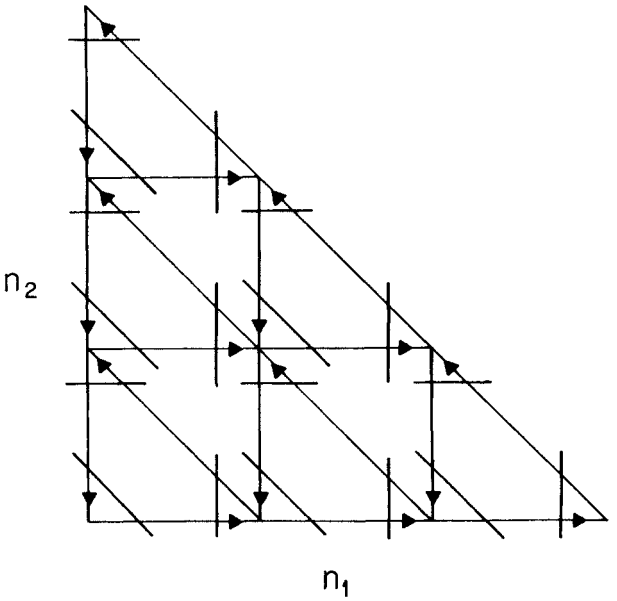

Fig. 3.5: Local Balance Pairings

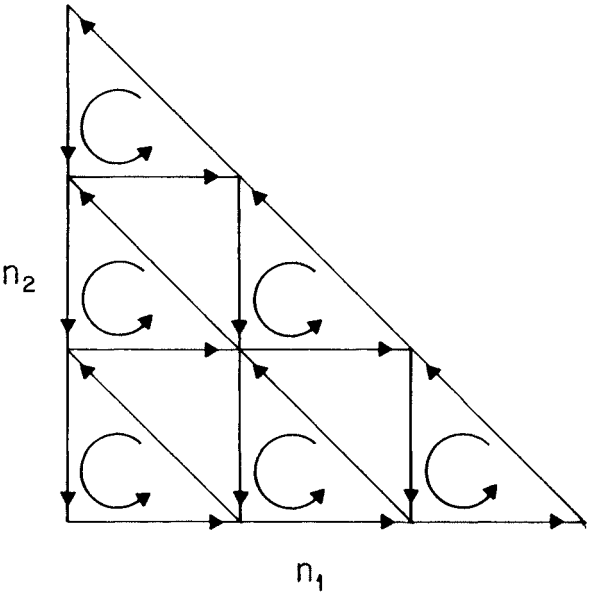

Fig. 3.6: Circular Flows

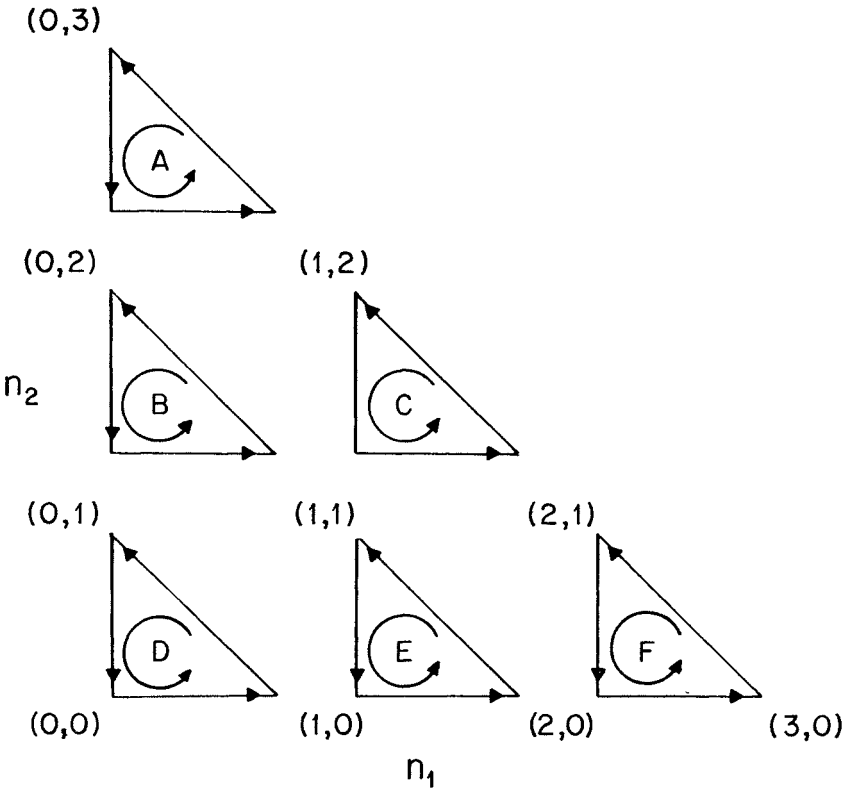

Fig. 3.7: Decomposed State Transition Diagram

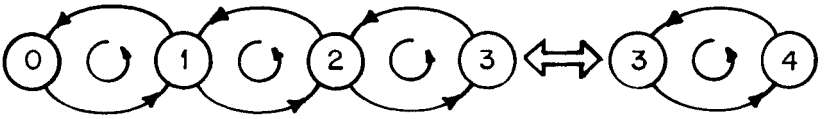

Fig. 3.8: M/M/1 System Building Blocks

and the solution:

$$p_{10} = \frac{\mu_1}{\mu_2} p_{00} \tag{3.42}$$

Substituting these solutions back into one another yields

$$p_{21} = \frac{\mu_1}{\mu_2}\frac{\mu_1}{\mu_3}\frac{\mu_1}{\mu_2} p_{00} = \left(\frac{\mu_1}{\mu_2}\right)^2 \left(\frac{\mu_1}{\mu_3}\right) p_{00} \tag{3.43}$$

or the product form solution. What we have done is solved the building blocks along a path from the desired state to the reference state. Actually this is a conservative system. Any path between the two states would have produced the same results.

If we solve for all the state probabilities we can deduce that:

$$p_{n_1 n_2} = \left(\frac{\mu_1}{\mu_2}\right)^{n_1} \left(\frac{\mu_1}{\mu_3}\right)^{n_2} p_{00} \tag{3.44}$$

One more point needs to be made. The state transition diagram of Figure 3.4 is the same as that for an open network of two queues in series, with an upper bound on the *total* number of customers in both queues. In the diagram the upper bound, N, takes the form of the boundary $n_1+n_2 \leq N$. As $N \to \infty$ and we go to an open network of two infinite buffer queues in series the state transition diagram becomes infinite in extent but everything that has been said about the building block structure still holds.

Example: Consider the (open) M/M/1 queueing system. The state transition diagram is constituted of building blocks consisting of two transitions each. For the *infinite* buffer M/M/1 queueing system there are an infinite number of such building blocks. In the case of the M/M/1/N *finite* buffer system, Figure 3.8, a finite number of such building blocks are pasted together. Note that the same state transition diagram models a (closed) cyclic queueing system of two queues.

Naturally there is only one path from any state to the usual reference state, p_0.

∇

Example: Consider a closed cyclic network of four queues with exponential servers and with only two customers in the network. The state

transition diagram is three dimensional and appears in Figure 3.9. The basic building block consists of four edges along a tetrahedron shaped volume of space. One could show that there are circular and equal valued flows of probability flux along the edges of these tetrahedrons and that any state probability can be found by solving these tetrahedrons along a path back to the reference probability.

Note that the state transition diagram of Figure 3.9 is the same as that for an open network of three queues in series, with an upper bound on the *total* number of customers in the queues. This bound takes the form of the plane boundary $n_1+n_2+n_3 \leq N$ in the state transition diagram. As $N \rightarrow \infty$ one has a system of infinite buffer queues and the state transition diagram consists of an infinite number of tetrahedral building blocks.

∇

We have seen that the state transition diagrams of a number of product form networks can be decomposed into elementary building blocks. In fact, as we shall see, this is true of any product form network. One can also look at the reverse case, the construction of state transition diagrams out of elementary building blocks. We will call the process of aggregating building blocks to form a state transition diagram, *geometric replication.* The reason is that the same geometric building block is usually replicated to form the overall diagram. Note that in the language of mathematical topology one would call a building block a "cell" and a state transition diagram would be called a "complex". Thus [LAZA 84a]:

Definition: The constructive process of aggregating multi-dimensional cells into a complex is referred to as *geometric replication.*

The geometric replication of cells or building blocks is useful in explaining the nature of the product form solution. It will also be useful later in devising models with this solution.

3.3.3 Building Block Circulatory Structure

In this section the ideas of the last section will be made more precise [WANG 86].

Definition: The *circulatory structure* of a state transition diagram is the pattern of probability flux on the diagram transitions.

Perhaps the most important property of building blocks for product form networks is that they can be embedded into the overall state transition diagram without affecting the flow pattern of the rest of the diagram, except for a renormalization. This property is what allows one to solve

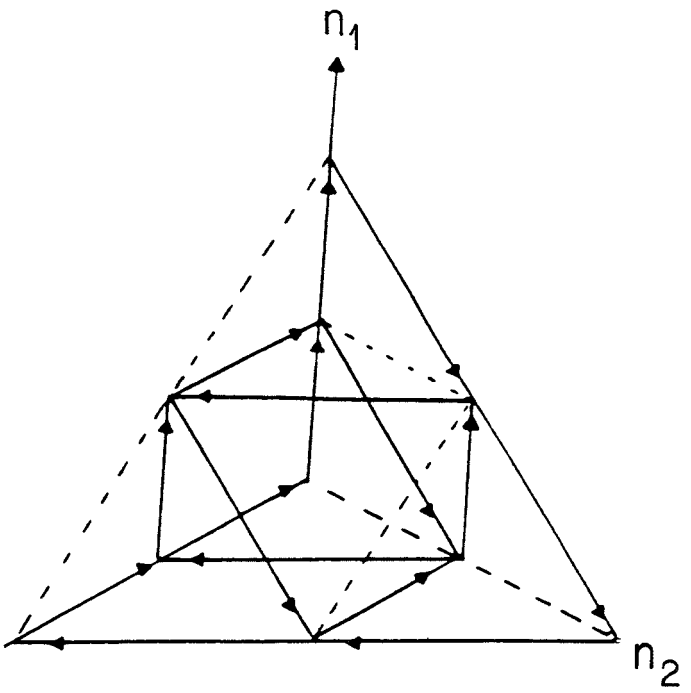

Fig. 3.9: State Transition Diagram for Four Queue Cyclic Network

building blocks in isolation in order to generate the recursive product form solution.

Definition: An *isolated circulation* consists of the probability flux along a subset of state transition diagram edges for which there is conservation of this flow at each adjacent state when the edges are embedded in the overall state transition diagram. Thus the relative equilibrium probabilities for these states can be solved for in isolation without considering the rest of the state transition diagram.

Definition: A *cyclic flow* is an isolated circulation in the form of a single cycle.

The "circular flows" of the building blocks of the previous section are cyclic flows. We can summarize by saying:

Duality Principle: There is a duality between the existence of local balance and the existence of isolated circulations. That is, the existence of local balance leads to isolated circulations and the presence of isolated circulations leads to local balance.

Proof: The former follows from the usual queueing interpretation of local balance. The latter follows from the definition of an isolated circulation.

∇

In other words, in observing local balance for a pair of transitions at a state, one is observing an isolated circulation passing through the state.

We will now characterize the building blocks:

Definition: A cyclic flow about L edges is said to be of length L.

Now let's recall the examples of the previous section for both open and closed networks. We can construct a table showing the lengths of the building blocks for the various cases:

Table 3.1: # of Queues vs. L

Closed	Open	L
2	1	2
3	2	3
4	3	4

This suggests the following:

Theorem: For a Markovian product form queueing network where the state of a queue is described by the number of customers in the queue, a closed path of L queues corresponds to an L length cyclic flow and an open path of L queues (into, through, and out of the network) corresponds to a L+1 length cyclic flow.

Proof: To prove this for a closed network is straight forward. Each queue in a closed path of L queues contributes one transition so the building block is of length L. For an open network of L queues each queue contributes one transition. There is also one additional transition that can be thought of as being due to a "virtual queue" that connects the output to the input. The transition rate of this additional transition is just the arrival rate to the network. This gives building blocks of length L+1.

∇

Next, let's characterize the sequence of events in a queueing network that are associated with, or serve to generate, a building block:

Definition a: For a closed network a *relay sequence* is a series of successive events consisting of a customer leaving queue i to enter queue j, followed by a departure from queue j which enters queue k, ... finally followed by an arrival back into queue i. Each of the queues is distinct and they form a closed cyclic path through the queueing network.

The sequence of events associated with a building block is called a relay sequence because a customer arriving at a queue is followed by a customer leaving the same queue. For a FIFO discipline this is not the same customer unless the queue is empty. It is as if customers are carrying a baton around a closed path, as in a relay race. There is an analogous definition for an open network:

Definition b: For an open network a "relay sequence" is a series of successive events corresponding to an arrival into queue i from outside the network, followed by a departure from queue i to queue j, followed by a departure from queue j to queue k, ... followed by a departure from the network. Each of the queues is distinct.

These ideas will now be expanded upon through a series of examples:

Example: Consider two queues, with exponential servers, in series forming an open network with Poisson arrivals, as in Figure 3.10. As has been mentioned, if the system has an infinite number of buffers the state transition diagram has building blocks as in Figure 3.7 but is infinite in

Fig. 3.10: Series (Tandem) Open Network

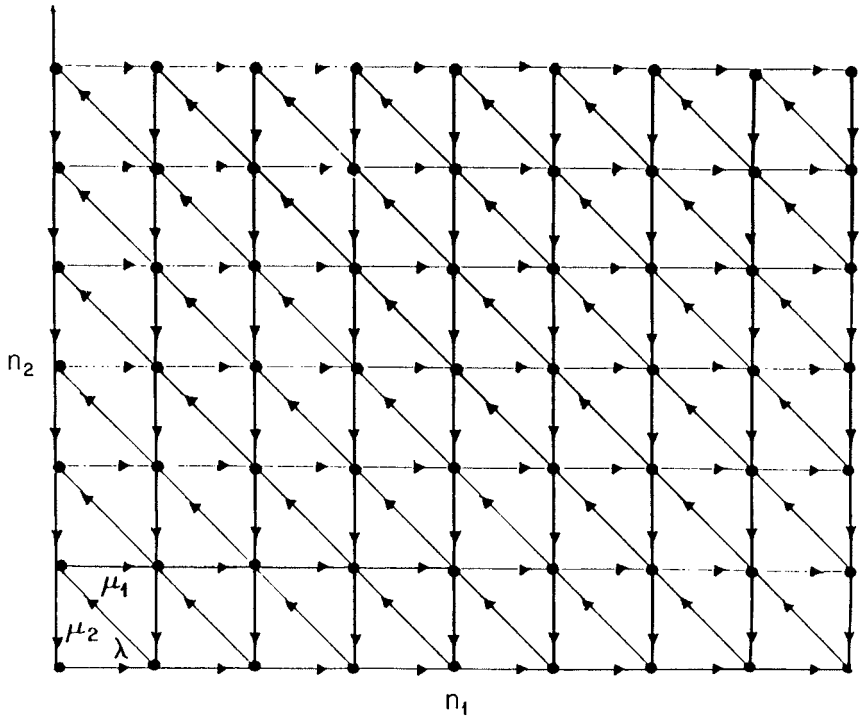

Fig. 3.11: Blocking State Transition Diagram

extent. The product form solution for this network is:

$$p_{n_1 n_2} = \left(\frac{\lambda}{\mu_1}\right)^{n_1}\left(\frac{\lambda}{\mu_2}\right)^{n_2} p_{00}$$

Suppose now that there is an upper bound on the number of customers in *each* queue: $n_1 \leq n_{1,max}, n_2 \leq n_{2,max}$ This gives rise to the state transition diagram of Figure 3.11 with vertical and horizontal boundaries. This blocking network is now a non-product form network.

Why is this so? What has happened is that there are no longer integral building blocks along the upper and right side rectilinear boundary. Put another way, the pairing of transitions for local balance has been disrupted along these boundaries. That is, there are transitions along the boundary that are missing their natural partner for local balance pairing. This deficiency along the boundary disrupts the flow of probability flux throughout the state transition diagram so that there are no longer cyclic flows about the remaining integral building blocks and the product form solution no longer exists. Nico van Dijk of the Free University, The Netherlands, has written [VAND] that there is a "failure" of local balance in such a case. At this point the only way to calculate the state probabilities directly is by numerically solving the set of linear global balance equations.

Not every boundary destroys the product form solution. For instance, suppose that we have an upper bound on the *total* number of customers in both queues. As has been mentioned this leads to the state transition diagram like that of Figure 3.4. This queueing network has the same product form solution, except for the normalizing value of p_{00}, as that for a similar infinite buffer network. What is different, compared to the blocking network, is that the diagonal boundary $n_1+n_2 \leq N$ allows integral building blocks and local balance pairing for all transitions, and thus the product form solution exists.

Finally, consider another modification where if either of the queues is full, customers will "skip" over it. The state transition diagram is shown in Figure 3.12. There are now integral building blocks of length two along the upper and right boundaries and integral building blocks of length three throughout the interior. Again there is local balance, cyclic flows and the same product form solution holds.

∇

Example: In this example we will look at the effect of random routing on the building block structure. The example involves a multiprocessor model.

A multiprocessor system [HWAN] consists of a group of processors interconnected with a group of memory modules. A closed Markovian queueing model of a multiprocessor system is shown in Figure 3.13. That

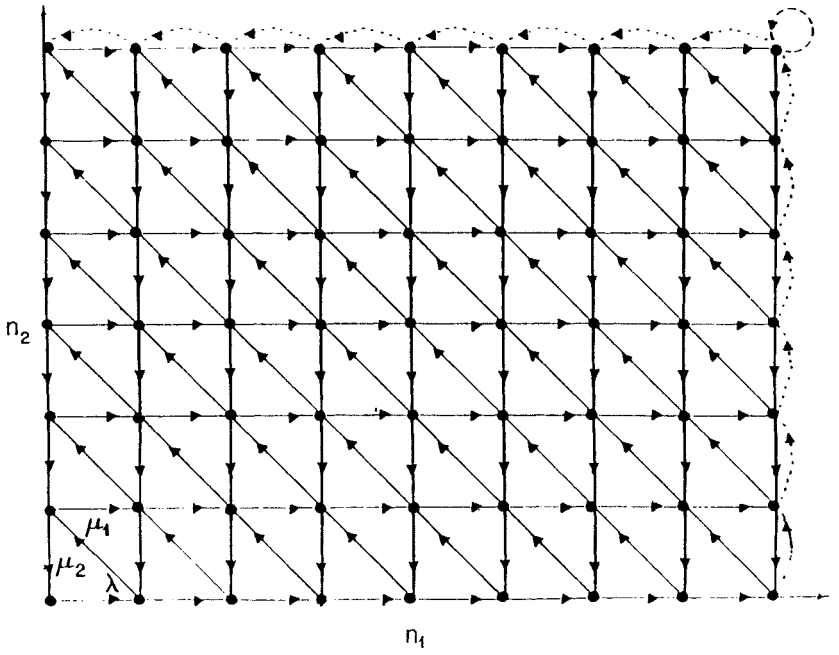

Fig. 3.12: Skipping State Transition Diagram

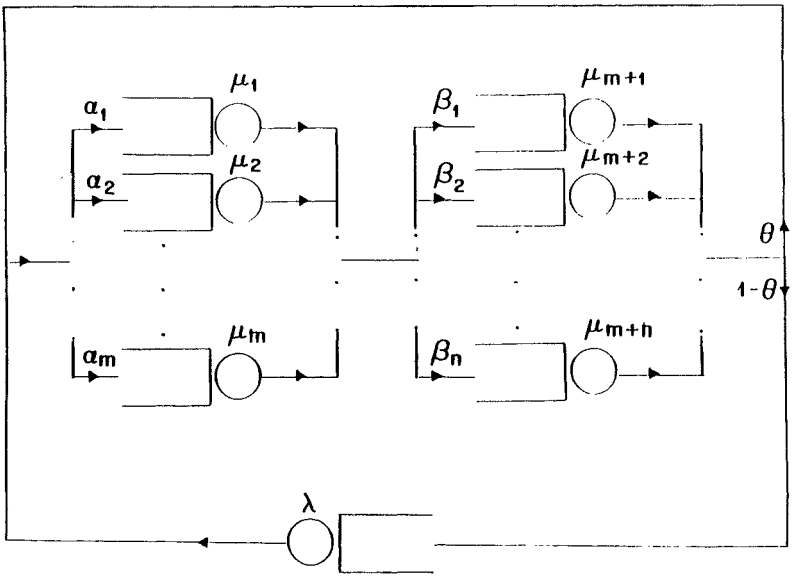

Fig. 3.13: Multiprocessor Queueing Model

is, the service times are exponential. From before, we know that this is a product form network. This model consists of three queueing systems arranged in a cyclic fashion. Two of the systems are comprised of a number of parallel queues and represent, respectively, the CPU's and memory modules. Routing into each of the two banks of parallel queues is random. That is, customers (really jobs) enter the m CPU's with probability $\alpha_1,\alpha_2, \cdots \alpha_m$ and customers enter the n memory modules with probabilities $\beta_1,\beta_2, \cdots \beta_n$. The third system is a single queue, departures from which represent submissions of jobs (or interactive commands) to the computer system.

In this multiprocessor system a customer leaving the memory module bank can proceed directly back to the CPU's with probability θ or it may go to the job submission queue to wait for its next execution with probability $1-\theta$. To simplify the following discussion we will concentrate on the special case when $\theta=0$.

For this reduced system, a "relay sequence" is a sequence of successive events corresponding to a customer entering a CPU queue followed directly by a customer departure from that CPU into a memory module followed directly by a customer departure from that module back to the job submission queue. Again, the departing customers are not necessarily the same as the arriving customers.

We will look at two types of building blocks, the second of which is a decomposed part of the first. The structure of the first building block for this m CPU and n memory module system is shown in Figure 3.14. Assume that "a" is a state $(k_1,k_2, \cdots k_l \cdots k_m,k_{m+1} \cdots k_{m+n})$ where k_l is the # of customers in the lth queue and the job submission queue is nonempty. A customer leaving the job submission queue corresponds to a transition into one of the 1,2 ... m CPU states, i.e., $(k_1,k_2, \cdots k_{l+1}, \cdots k_m,k_{m+1}, \cdots k_{m+n})$. A customer leaving a CPU corresponds to a transition into one of the m+1, m+2 ... m+n memory module states. Finally a memory module departure corresponds to a transition back to state a.

Thus, in queueing terms, this first building block is generated by *all possible* relay sequences starting from state a. The building block's net flow into and out of the block's states balance when the block is embedded in the state transition diagram. The building block is a subgraph whose relative state probabilities can be solved for in isolation.

Now let us consider the second type of building block. Let p_l be the state probability that the customer is in the CPU queue l if $1 \leq l \leq m$ and in memory module l if $m+1 \leq l \leq m+n$. Using flux conservation at a CPU state

$$\alpha_l \lambda p_a = \sum_{j=1}^{n} \beta_j \mu_l p_l = \mu_l p_l \qquad l=1,2,3...m \tag{3.45}$$

and at a memory module state:

$$\mu_l p_l = \beta_{l-m} \sum_{i=1}^{m} \mu_i p_i \qquad l=m+1,m+2,\ldots m+n \tag{3.46}$$

Solving these for p_l yields:

$$p_l = \frac{\alpha_l \lambda}{\mu_l} p_a \qquad l=1,2,3\ldots m \tag{3.47}$$

$$p_l = \frac{\beta_{l-m} \lambda}{\mu_l} p_a \qquad l=m+1,m+2,\ldots m+n \tag{3.48}$$

The circulatory structure of the building block of Figure 3.14 can be decomposed into an aggregation of smaller cyclic flows which are associated with smaller building blocks. For instance, $\alpha_i \lambda$, the flux from state a to state i, where $1 \leq i \leq m$, can be decomposed as $\sum_{j=1}^{n} \alpha_i \beta_j \lambda$ in which $\alpha_i \beta_j \lambda$ is the portion of flow which corresponds to departures from the ith CPU destined for the jth memory module. The flow corresponding to departures from memory module j can also be decomposed into sub-flows. These are proportional to $\alpha_1, \alpha_2, \cdots \alpha_m$.

The circulatory structure of the first building block of Figure 3.14 with its splits and joins of flux can thus be decomposed into an aggregation of simple cyclic flows embedded along smaller building blocks. Each such flow corresponds to a *specific* relay sequence originating from state a. One such cyclic flow/building block is shown in Figure 3.15. It illustrates a relay sequence with m=n=3 involving the 2nd CPU and the 1st memory module.

These cyclic flow building blocks can also be solved in isolation for the equilibrium probabilities. They are pasted together along edges to form the larger building block of Figure 3.14. They form the most basic decomposition possible of the overall circulatory pattern of the multiprocessor state transition diagram with this state description.

∇

Example: Consider now a different queueing system example. This is a FIFO queue processing two different classes of customers. The server is exponential with rate μ for both classes. The state consists of the class of the job in each queue position. The state transition diagram of this queueing system is shown in Figure 3.16. The numbers associated with each state represent the class of each job in each queueing system position. In

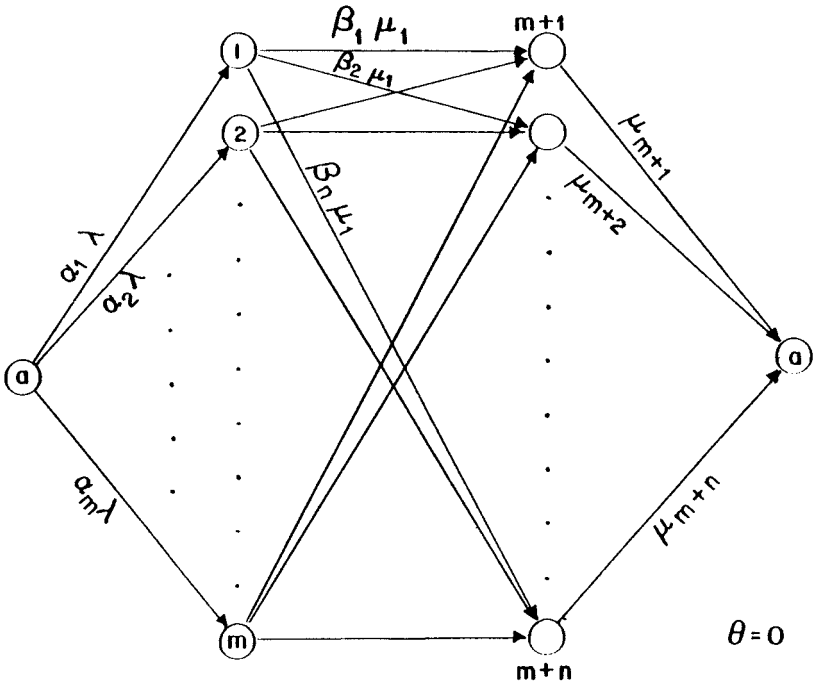

Fig. 3.14: First Multiprocessor Building Block

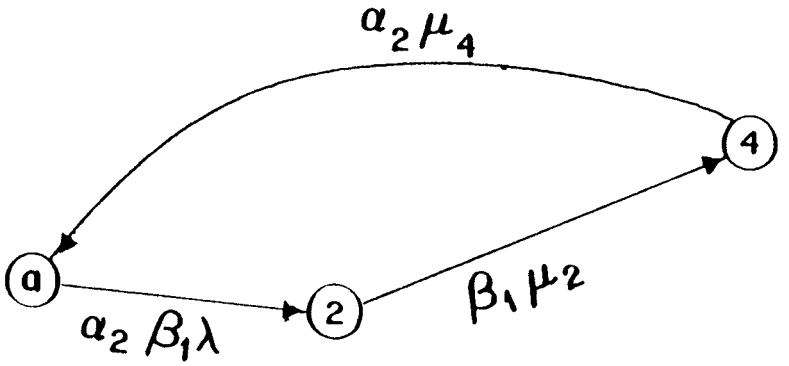

Fig. 3.15: Second Multiprocessor Building Block

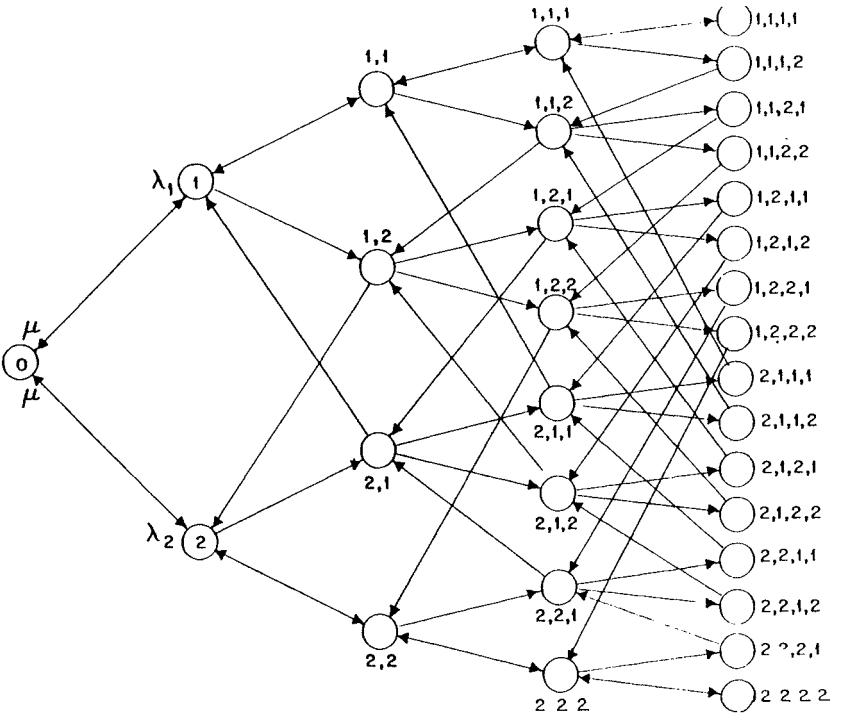

Fig. 3.16: FIFO Single Queue State Transition Diagram

the diagram's notation they are served from left to right.

This is a product form network. There are cyclic flows in the state transition diagram. In terms of queueing system events these correspond to a shift register like behavior:

Definition: A "shift register sequence" for a FIFO multiclass product form queue where the state involves the customer class at each queue position is a sequence of successive queueing events corresponding to a class i departure followed by a class i arrival followed by a class j departure (j may =i) followed by a class j arrival, and so on, terminating when the queue returns to the original state. This sequence of events generates the state transition diagram building blocks.

For instance the state transition sequence:

$$121 \rightarrow 21 \rightarrow 211 \rightarrow 11 \rightarrow 112 \rightarrow 12 \rightarrow 121$$

corresponds to a cyclic flow of length six. The state transition diagram of this queueing system consists of cyclic flows of different lengths. There is a group of local balance equations for each cyclic flow which can be solved without regard to the rest of the state transition diagram.

Let stage i=1,2... in the diagram consist of the transitions between states with i and i-1 total customers in those states. The cyclic flows exist in a single stage. Moreover, there are cyclic flows of different lengths within the same stage (Table 3.2). This is because the shift register like behavior may return the system to a starting state with a number of shifts less than the queue length (consider 1212).

Table 3.2: Number of Cyclic Flows

	Stage					
Circulation Length	1	2	3	4	5	6
2	2	2	2	2	2	2
4	-	1	-	1	-	1
6	-	-	2	-	-	2
8	-	-	-	3	-	-
10	-	-	-	-	6	-
12	-	-	-	-	-	9
Total	2	3	4	6	8	14

If n_1 is the number of class one customers in the queueing system and n_2 is the number of class two customers in the queueing system then:

$$p(n_1,n_2) = \frac{\lambda_1^{n_1}\lambda_2^{n_2}}{\mu^{n_1+n_2}} p(0,0)$$

Note that in having the state of the system indicate the queueing order of the two classes, one has a more detailed decomposition of the state transition diagram then that when the state indicates only the total number of each class at a queue. The cyclic flows in the state transition diagram of Figure 3.16 are a decomposition of isolated circulations in the diagram where state does not indicate queueing order.

For networks of such FIFO queues the basic geometric building block is also generated by shift register like sequences. As an example, consider the cyclic two queue system of Figure 3.17. The state transition diagram is shown in Figure 3.18 for four customers. Here μ_1^{II}, for instance, indicates the service rate of the 1st queue for class II customers. To obtain a product form solution, service rates must be class independent. Then

$$p(n_1,n_2) = \left(\frac{\mu_2}{\mu_1}\right)^{n_1+n_2} p(0,0)$$

where n_1 and n_2 are the number of class I and class II customers in the upper queue, respectively.

Cyclic flows exist between adjacent columns of states in the diagram of Fig. 3.18. One such cyclic flow corresponds to the state transition sequence:

$$1,122 \rightarrow 11,22 \rightarrow 1,221 \rightarrow 12,21 \rightarrow 2,211 \rightarrow 22,11 \rightarrow 2,112 \rightarrow 21,12 \rightarrow 1,122$$

where the state of the upper queue appears at left.

Note that the sequence starts with a class 1 customer leaving the lower queue followed by a class 1 departure from the upper queue. While this is reminiscent of a relay sequence, one has not returned to the original state since the customer order in both queues is changed. The sequence thus continues cycling until it returns to the original state. It is as if a shift register sequence, in the context of a queueing network, is comprised of successive relay sequences. Let's make precise the sequence of events that generate the state transition diagram building blocks.

Definition: For a closed FIFO product form network in which the state indicates the class of customer at each queue position a "shift register sequence" is a series of successive queueing events beginning with a customer leaving queue i to enter queue j, followed by a departure from queue j which enters queue k... followed by an arrival back into queue i. If the

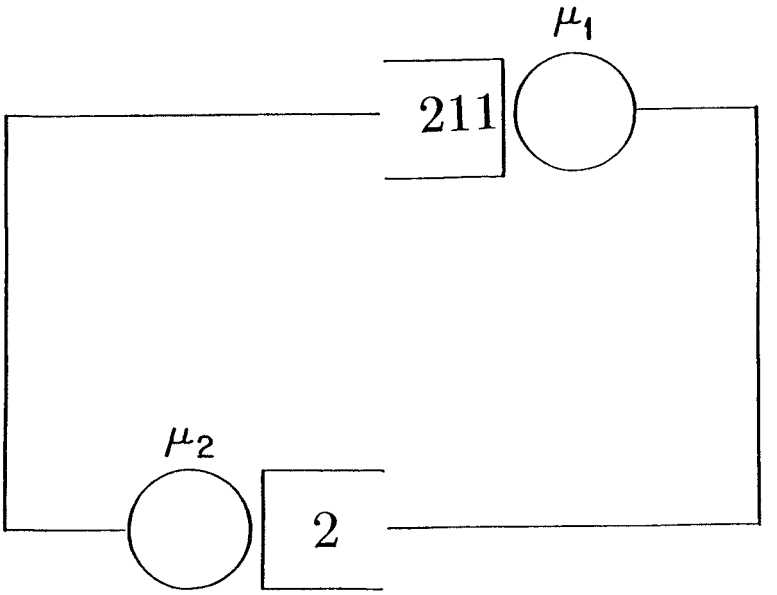

Fig. 3.17: Closed FIFO Network

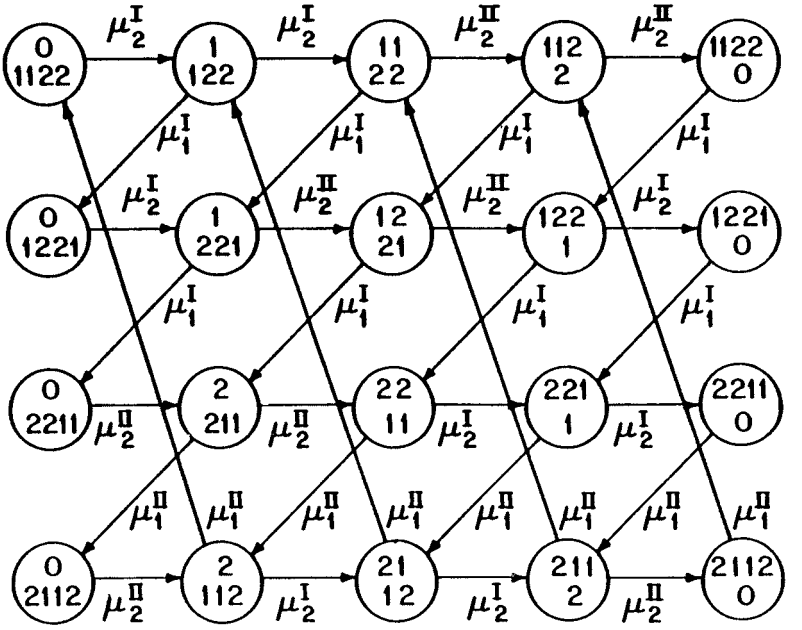

Fig. 3.18: Closed FIFO Network State Transition Diagram

network is not now in the original state, the previous series of events repeats until the network returns to the original state. Again, each of the queues is distinct and they form a closed cyclic path through the queueing network.

Definition: For an open FIFO network in which the state indicates the class of customer at each queue position a "shift register sequence" is a series of successive events beginning with an arrival into queue i from outside of the network followed by a departure from queue i to enter queue j, followed by a departure from queue j which enters queue k... followed by a departure from the network. If the network is not now in the original state, the previous series of events repeats until the network, returns to the original state. In the sequence, network departures of one class are followed by network arrivals of the same class. Each of the queues is distinct.

Theorem: For a FIFO product form queueing network where the state is described by the class of each customer in each queue position, and for which there is a consistent (see below) set of local balance equations, a closed path of N queues corresponds to a cyclic flow of length M ($M \leq N$) multiplied by the total number of customers in the path. An open path of N queues corresponds to a cyclic flow of length M ($M \leq N+1$) multiplied by the total number of customers in the path.

Proof: The cyclic flow of greatest length occurs when each customer must be shifted completely around the cyclic path of queues (N for a closed path and N+1 for an open path with returns through a virtual queue). For certain states customers may not have to be shifted around the entire path to bring the system back to the original state. This accounts for the inequalities in the Theorem.

∇

For networks of LCFS queues in which the state indicates the class of customer at each queue position the basic geometric building block is generated by a relay type sequence. The difference is that, because of the LCFS discipline, the job departing the first queue in a closed path is the *same* job that eventually returns. For an open path a job of the same class returns.

3.3.4 The Consistency Condition

Recall that the approach adopted in the previous sections calls for decomposing the state transition diagram into its building blocks and for solving the associated local balance equations. In the two dimensional case, we have noted the existence of distinct paths connecting some arbitrary

states with the reference state. This property is not observed in the one dimensional case since the state transition diagram is a tree and hence a unique path links any node to the reference solution.

Thus the problem of consistency of the set of local balance equations arises as a result of the existence of distinct paths from an arbitrary state to the reference state. If, by taking alternate paths, the equilibrium probabilities for the same state are different, the set of local balance equations is not consistent.

As in the lower dimensional case the set of global balance equations must be decomposed into a set of local balance equations. Such a decomposition is, however, not unique. In addition it is not clear whether the resulting set of local balance equations is consistent. If the local balance equations are consistent, however, they are equivalent with the global balance equations. This is because the latter has a unique solution.

To derive the necessary and sufficient conditions for the consistency of a set of local balance equations, we define the following *consistency graph* [LAZA 84C]:

Definition: A consistency graph is an oriented graph, topologically equivalent with the original state transition diagram. In addition, it has the property that the probability at each node is equal to the probability of any adjacent node multiplied with the value of the associated edge connecting the two nodes.

The class of consistency graphs of interest to us derives the algebraic values associated with the arcs directly from the set of local balance equations. Thus, the consistency graph can be seen as an "easy to read" graphical representation of the *proposed solution* for the local balance equations.

Example: Consider the state transition diagram of Figure 3.4 for the three queue cyclic Markovian queueing network of Figure 3.3. The consistency graph for this system appears in Figure 3.19. It is the consistency graph for what is, at this point, a *proposed* solution and associated local balance equations. For instance, the relationship between p_{21} and p_{11} is:

$$p_{21} = \frac{\mu_1}{\mu_2} p_{11}$$

Now choose any path (open or closed) on the consistency graph. Associated with it is an algebraic value called the *product* of the edges or simply the product. For example, with the path from state (0,0) to (0,1) to (1,1) to (1,2) is associated the product:

$$\frac{\mu_1}{\mu_3} \frac{\mu_1}{\mu_2} \frac{\mu_1}{\mu_3}$$

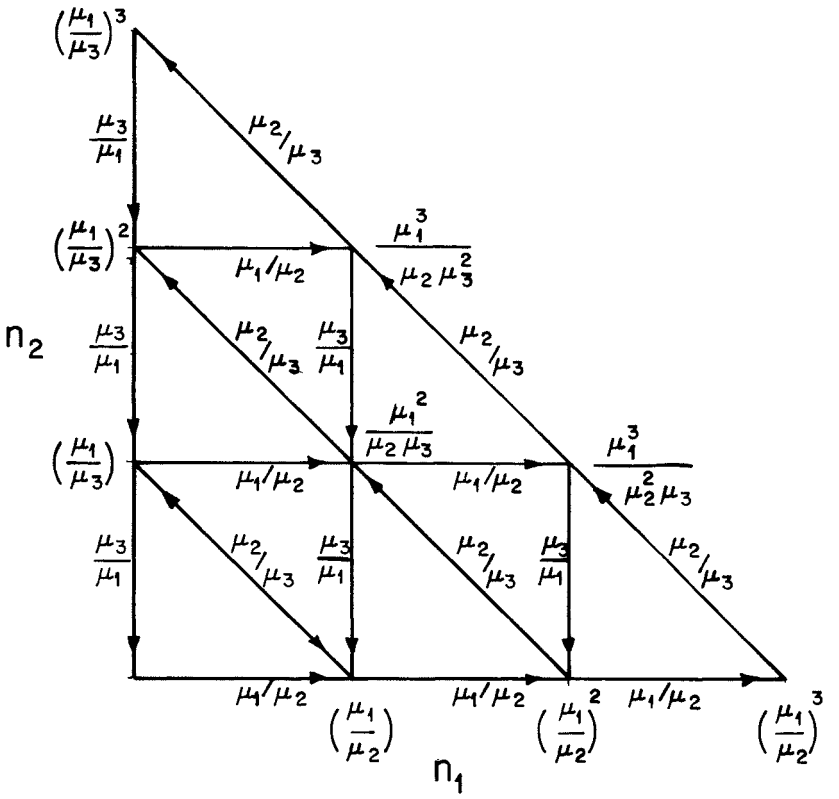

Fig. 3.19: Consistency Graph for Fig. 3.4.

∇

Note that reversing the orientation of an arc of the consistency graph with value a results in an arc with value a^{-1}. We can now state the following consistency theorem [LAZA 84c]:

Theorem: A system of local balance equations is consistent if and only if any closed path of the consistency graph has the product equal to one.

Proof: Assume that there is a closed path in the consistency graph such that its product is not equal to one. Therefore, for at least one node belonging to this path, there are two distinct paths leading to the reference node which do not have the same product. Thus two different values for the probability associated with such a node can be obtained. Hence, the local balance equations are not consistent.

The sufficiency part of the theorem can be shown by direct computation. The product of an arbitrary path connecting node $(n_1,n_2,n_3, \cdots n_m)$ with the reference node $(0,0,0, \cdots 0)$ is an algebraic invariant. This is because, by assumption, the product of any closed path is equal to one and reversing the orientation of any edge with value a results in an edge with value a^{-1}. The probability at node $(n_1,n_2,n_3, \cdots n_m)$ is, therefore, equal to the product of the path connecting this node with the reference node times the probability of the reference node.

∇

Hence to determine the consistency of the set of local balance equations, the product of any closed path of the consistency graph has to be computed and compared with the value one. This will be referred to as the *consistency condition.* Naturally, the consistency condition is redundant for some closed paths. The minimum set of products needed to verify the consistency condition is investigated from a topological point of view in [LAZA 84c].

It is easy to verify for Figure 3.19 that the product of any closed path is one and thus there is a consistent set of local balance equations, a building block decomposition and a product form solution.

3.4 Recursive Solution of Non-Product Form Networks

3.4.1 Introduction

In terms of applications, there are many important and useful non-product form models. In general, these do not have closed form solutions for their equilibrium probabilities. However quite a few models are amenable to setting up a series of recursive equations to determine their

equilibrium probabilities.

The idea of using recursions was first discussed in a systematic way by Herzog, Woo and Chandy [HERZ][SAUE 81]. To date such recursions have usually been developed for relatively simple systems with one or two queues. However there has been some work on algorithmically determining such recursions [PARE] [ROBE 89].

In what follows two classes of non-product form models for which recursive, or at least decomposed, solutions are possible will be presented.

In [LAZA 87] a class of non-product form networks is described whose state transition diagrams can be shown to be equivalent to a lattice tree of simplexes. In this "flow redirection" method the state transition diagram geometry is manipulated by equivalence transformations. Sequential decomposition refers to the related process of solving one subset of states at a time for the equilibrium probabilities:

Definition: A solvable subset of queueing network states is a subset of states for which the equilibrium probabilities can be determined without regard to the equilibrium probabilities of the remaining unknown states. Here probabilities are determined with respect to a reference probability.

Note that the direct solution of linear equations takes time proportional to the cube of the number of equations. If P states can be solved as S subsets then the computational effect is proportional to $(P/S)^3 * S$ rather than P^3.

Two types of structure which allow the state transition diagram to be decomposed into solvable subsets will now be presented. The first type of structure [ROBE 89], is illustrated in Fig. 3.20. Here each circular subset represents a state or a group of states. For the ith subset the rule is that there must be only one state external to the subset, with *unknown* probability, from which a transition(s) entering the subset originates. The subsets are solved sequentially, starting from the first subset to the second and so on. There is no restriction on the number of transitions which may leave the ith subset for destinations in the j=i+1,i+2,... subsets. This type of structure will be referred to as type A structure.

The second type of structure is illustrated in Fig. 3.21. Here the first subset consists of a single state. The remaining subsets each consist of a state or a group of states. These are arranged in a tree type of configuration with the flow between subsets from the top of the diagram to the bottom and a return flow from the bottom level back to the top level. The subsets may be solved from the top to the bottom. Transitions may traverse several levels (e.g. dotted line in the figure) as long as the direction of flow is downward. This type of structure will be referred to as Type B structure.

It is possible to write recursive equations for the equilibrium probabilities when each subset in the Type A or Type B structure consists of a

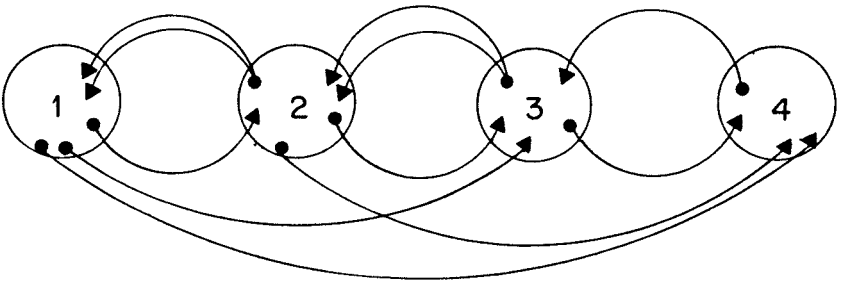

Fig. 3.20: Type A Structure

Copyright 1990 IEEE

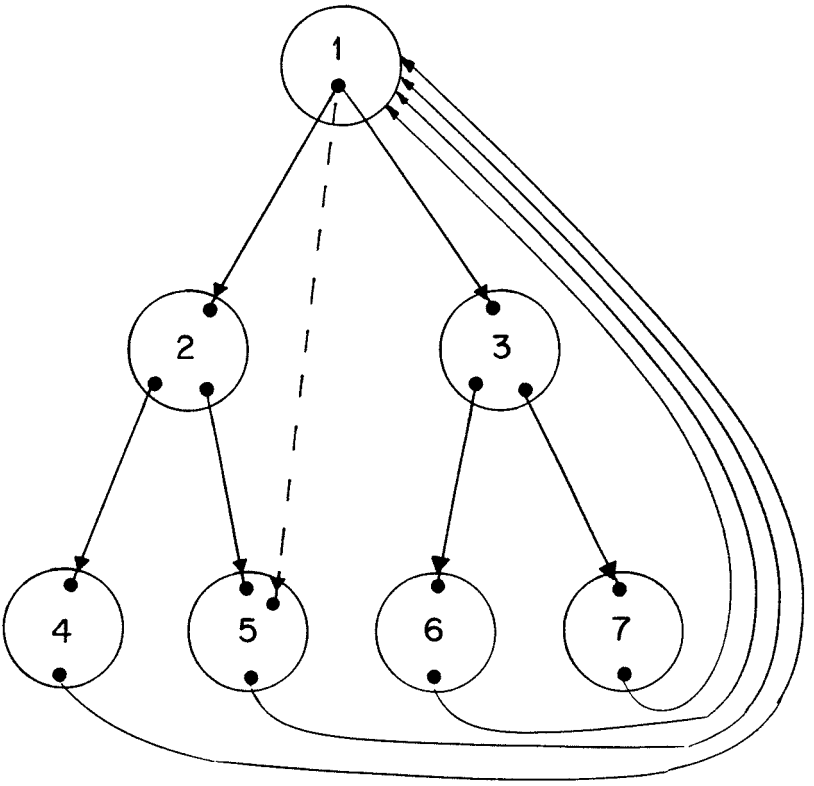

Fig. 3.21: Type B Structure

Copyright 1990 IEEE

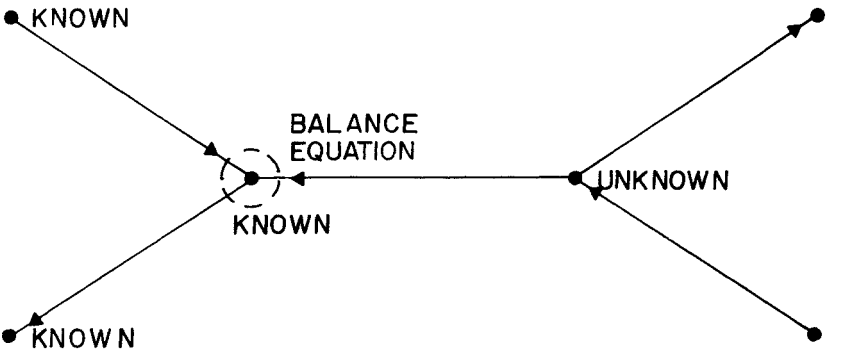

Fig. 3.22: Type A Structure: One State Per Subset

Copyright 1989 IEEE

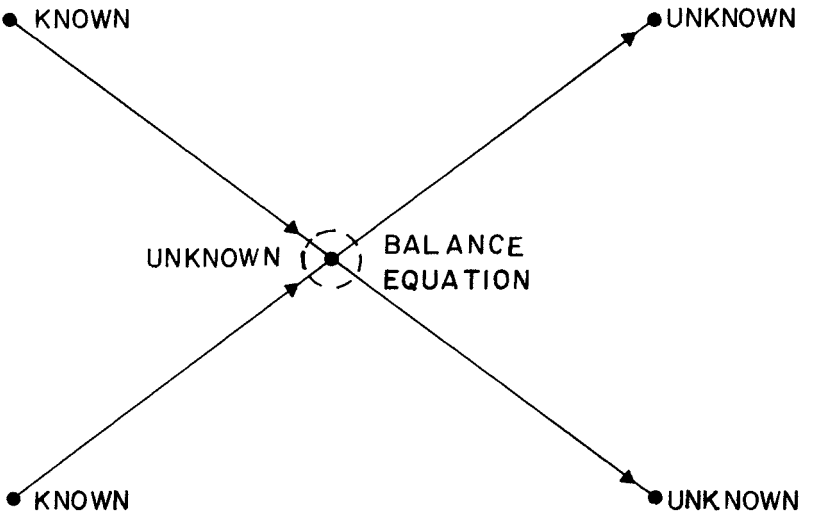

Fig. 3.23: Type B Structure: One State Per Subset

Copyright 1989 IEEE

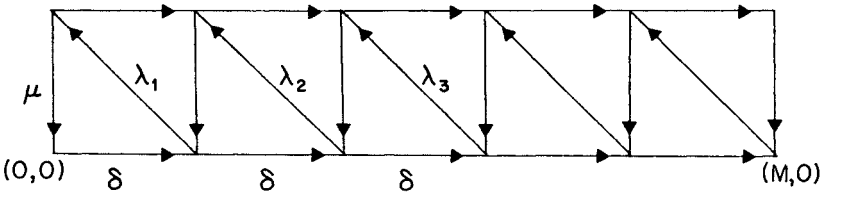

Fig. 3.24: Tandem Queues: Buffer Size 1 for 2nd Queue

Copyright 1990 IEEE

single state. In order to understand how recursive solutions are produced it is helpful to take a closer look at the Type A and B structures when one state at a time is decomposed. Figure 3.22 illustrates how a solution is arrived at for the Type A structure. A balance equation is solved for a state whose probability is already known. The state must have only one incoming transition from another state with unknown probability. When the balance equation is solved this other state probability becomes known. Often, this other state provides the next balance equation for solution.

Figure 3.23 illustrates how a solution is arrived at for the Type B structure. A balance equation is solved for a state with unknown probability where all incoming transitions originate from states with known probabilities.

A number of examples of the recursive solution of practical queueing networks follows. They are from [WANG].

3.4.2 Recursive Examples

A. Two Blocking Tandem Queues

This first example concerns two queues in tandem with finite buffers. The first queue has a buffer size of M and the second queue has a buffer size of one. Here δ is the system arrival rate and λ_i and μ are the service rates of the first and second queues, respectively. The arrival rate of the first queue is state dependent. As with all examples in this correspondence, the arrival process is Poisson and service rates follow an exponential distribution. Recursive equations for this case appear below. The state transition diagram is illustrated in Fig. 3.24.

Initialize:

$$p(0,0)=1.0 \qquad (3.49)$$
$$p(0,1)=(\delta/\mu)p(0,0)$$

Iterate: i=1,...,M-1

$$p(i,0)=(\delta/\lambda_i)[p(i-1,0)+p(i-1,1)]$$
$$p(i,1)=(\delta/\mu)[p(i-1,1)+p(i,0)]$$

Terminate (Right Boundary):

$$p(M,0)=(\delta/\lambda_M)[p(M-1,0)+p(M-1,1)]$$

$$p(M,1)=(\delta/\mu)p(M-1,1)$$

These equations have a simple form. The structure is Type A. It should be noted that the equations are not unique. It is also possible to take state dependent arrival rates into account for this and the remaining examples. Recursive equations for the case when the second queue's buffer is of size two appears below. Here μ_i is the state dependent service rate of the second queue. The state transition diagram is illustrated in Fig. 3.25.

Initialize:

$$\begin{aligned} p(-1,2)&=0.0 \\ p(0,0)&=1.0 \\ p(0,1)&=(\delta/\mu_1)p(0,0) \end{aligned} \tag{3.50}$$

Iterate: i=1,2,...,M-1

$$p(i,1)=\delta f(i)/g(i)$$

where

$$\begin{aligned} f(i)&=(\delta+\lambda_i)(\delta+\mu_2)[p(i-1,0)+p(i-1,1)] \\ &+\delta(\delta+\lambda_i)p(i-2,2)-\lambda_i(\delta+\mu_2)p(i-1,0) \\ g(i)&=\lambda_i[\mu_1(\delta+\mu_2)+(\delta+\lambda_i)(\delta+\mu_2)-\delta(\delta+\lambda_i)] \end{aligned}$$

$$\begin{aligned} p(i,0)&=[1/(\delta+\lambda_i)][\delta p(i-1,0)+\mu_1 p(i,1)] \\ p(i-1,2)&=[1/(\delta+\mu_2)][\lambda_i p(i,1)+\delta p(i-2,2)] \end{aligned}$$

Terminate (Right Boundary):

$$p(M,1)=\delta f'(M)/g'(M)$$

where

$$\begin{aligned} f'(M)&=\lambda_M(\delta+\mu_2)p(M-1,1)+\delta\lambda_M p(M-2,2) \\ g'(M)&=\lambda^2{}_M\mu_2+\lambda_M\mu_1(\delta+\mu_2) \end{aligned}$$

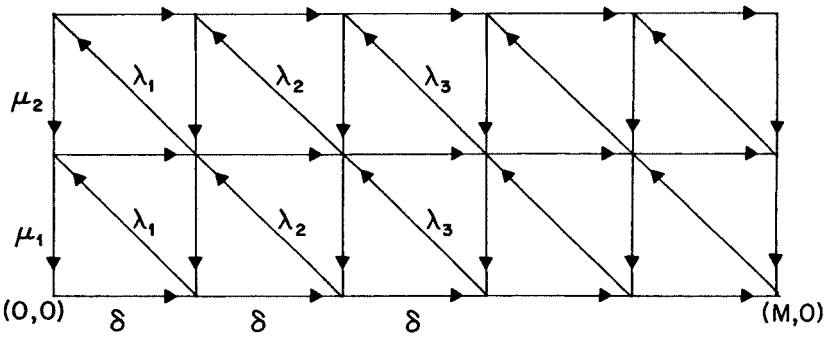

Fig. 3.25: Tandem Queues: Buffer Size 2 for 2nd Queue

Copyright 1990 IEEE

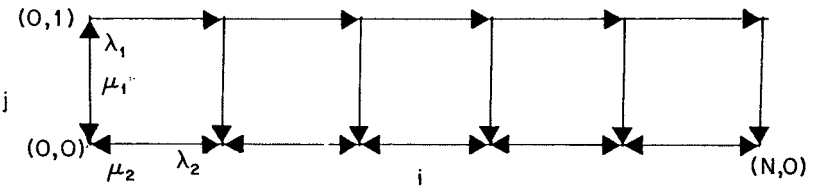

Fig. 3.26: ISDN State Transition Diagram

Copyright 1990 IEEE

$$p(M,0)=[1/(\lambda_M)][\delta p(M-1,0)+\mu_1 p(M,1)]$$
$$p(M-1,2)=[1/(\delta+\mu_2)][\lambda_M p(M,1)+\delta p(M-2,2)]$$
$$p(M,2)=(\delta/\mu_2)p(M-1,2)$$

Again, the structure is Type A. The first subset consists of (0,0). Then subsets consist of the states (i-1,1), (i-1,2) and (i,0) for i=1,2...M. At the right boundary the last subset consists of (M,1) and (M,2). It appears that these equations can not be extended to a larger buffer size for the second queue because of the resulting geometry of the state transition diagram. The case where the first queue's buffer is of size one or two and the second queue's buffer is of size M can be handled through a straightforward change of variables.

B. An ISDN Protocol

Consider a channel that can service either voice or data transmissions [SCHW 87]. The transmitter contains a buffer of size N for data packets. A voice call can seize the channel with rate λ_1 if no data packets are in the buffer. It is serviced at rate μ_1. If a voice transmission has seized the channel, then data packets can not be serviced but are buffered. Data packets arrive at rate λ_2 and are serviced at rate μ_2. The state transition diagram appears in Fig. 3.26. Interestingly the upper row of the diagram has the Type B structure and the lower row has the Type A structure. The recursive equations appear below. We note that the recursive expression for the row 1 appears in [SCHW 87].

Initialize:

$$p(0,0)=1.0 \tag{3.51}$$
$$p(0,1)=[\lambda_1/(\lambda_2+\mu_1)]p(0,0)$$

Iterate: i=1,...,N-1

$$p(i,0)=(\lambda_2/\mu_2)[p(i-1,0)+p(i-1,1)]$$
$$p(i,1)=[\lambda_2/(\lambda_2+\mu_1)]p(i-1,1)$$

Terminate (Right Boundary):

$$p(N,0)=(\lambda_2/\mu_2)[p(N-1,0)+p(N-1,1)]$$
$$p(N,1)=(\lambda_2/\mu_1)p(N-1,1)$$

C. A Window Flow Control Protocol

Consider the queueing system of Fig. 3.27 which models a window flow protocol [SCHW 87]. Packets are sent from the lower queue (source) through a channel (upper queue) to the destination (W box). The W box only releases packets when all W packets in the closed queueing network are in the W box. This models a reception acknowledgement of the entire group of packets. The state transition diagram appears in Fig. 3.28. The structure is type B. The recursive equations are:

Initialize:

$$p(0,0)=1.0 \tag{3.52}$$

Iterate: i=1,...,W-1

$$p(i,0)=[\lambda/(\lambda+\mu_i)]p(i-1,0)$$
$$p(W,0)=(\lambda/\mu_W)p(W-1,0)$$

Iterate: j=1,...,W-1

$$p(0,j)=(\mu_1/\lambda)p(1,j-1)$$

Iterate: i=1,...,W-j-1

$$p(i,j)=[(1/(\mu_i+\lambda)][\lambda p(i-1,j)+\mu_{i+1}p(i+1,j-1)]$$
$$p(W-j,j)=$$
$$(1/\mu_{W-j})[\lambda p(W-j-1,j)+\mu_{W-j+1}p(W-j+1,j-1)]$$

3.4.3 Numerical Implementation of The Equations

Scaling problems are possible with the use of some of these equations. As an example, consider equation (3.49). The first queue may be viewed approximately as an M/M/1 queue. The equilibrium probability at the ith state is equal to λ/μ times the equilibrium probability at the (i-1)th state. Here, λ is the arrival rate, and μ is service rate. Thus, for instance, the probability that there are one hundred packets in the buffer is $(\lambda/\mu)^{100}$ times the probability that the buffer is empty. This can naturally lead to numerical scaling and underflow problems.

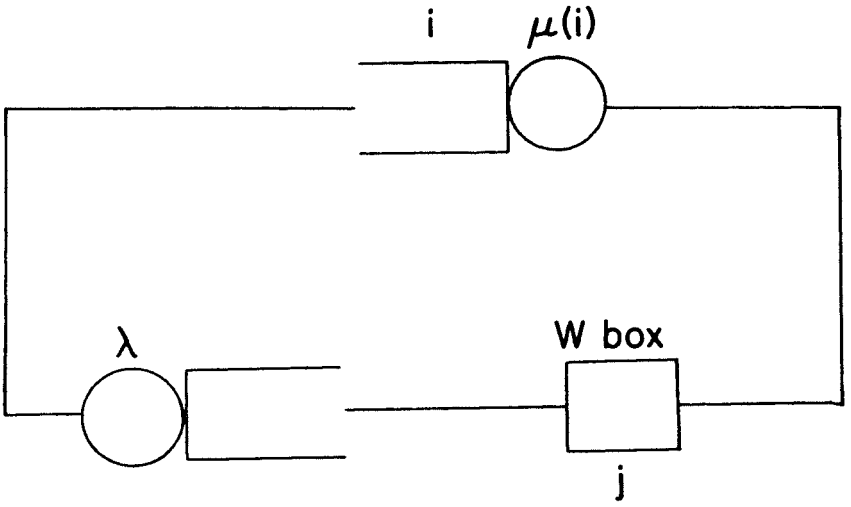

Fig. 3.27: Window Flow Control Model
Copyright 1990 IEEE

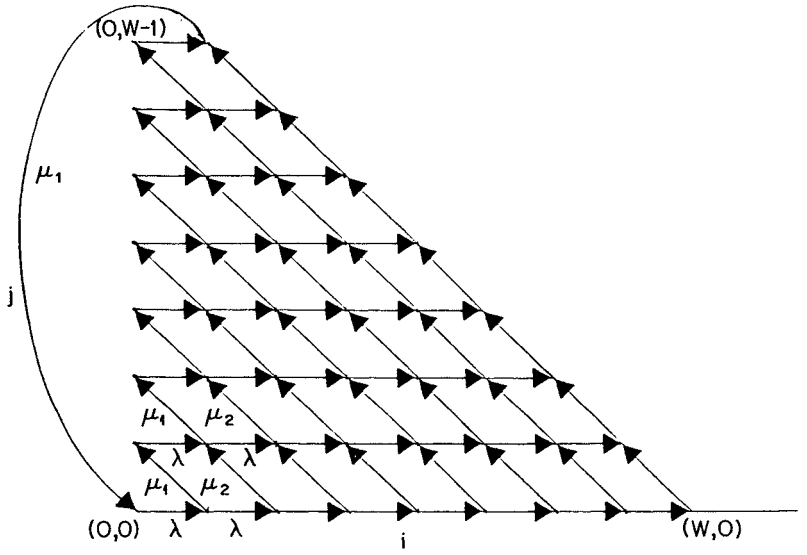

Fig. 3.28: State Transition Diagram of Fig. 3.27
Copyright 1990 IEEE

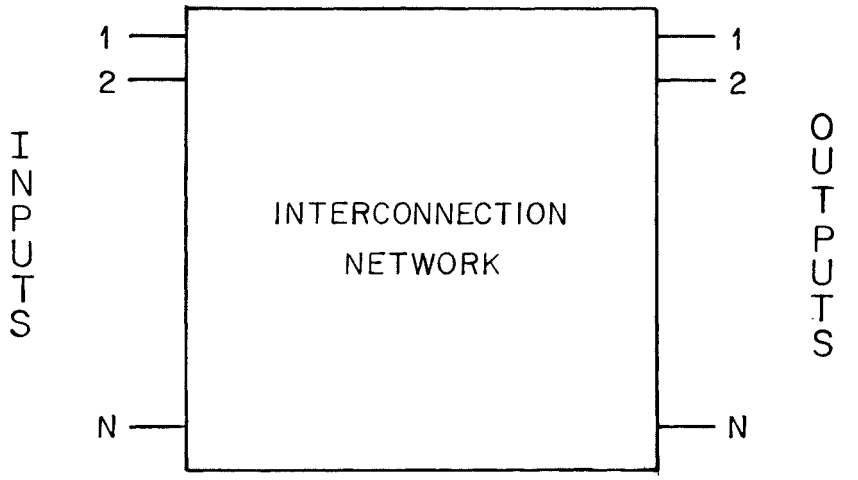

Fig. 3.29: Interconnection Network

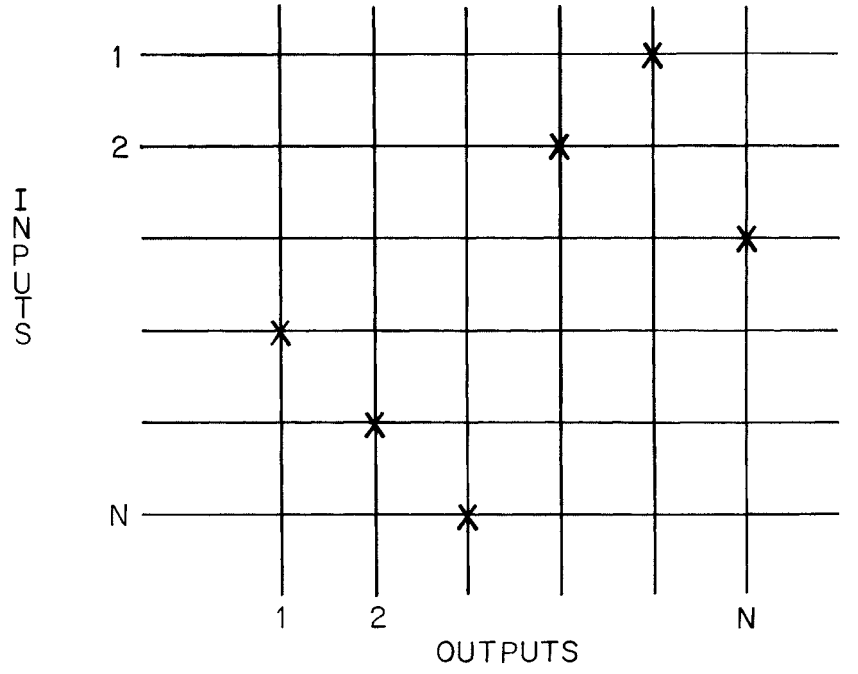

Fig. 3.30: NxN Crossbar Switch

One form of automatic scaling is possible in this if it is clear which section of the state transition diagram corresponds to light traffic and which corresponds to heavy traffic. This is the case for the linear state transition diagram associated with equation (3.49). The equilibrium probabilities are then calculated in this roughly monotonic order. When the size of equilibrium probability begins to approach the computer register size, it is scaled downward by a factor which brings the equilibrium probability down to the lower limit of machine register size.

3.5 Case Study I: Queueing on a Space Division Packet Switch

3.5.1 Introduction

As long as there have been telephones, and more recently with multiprocessor systems, a basic problem has been the interconnection of N inputs and N outputs. That is, how does one design a box, as in Figure 3.29, that simultaneously provides connection paths from the multiple inputs to multiple outputs. In traditional telephony the paths are actual circuits through the interconnection network (or simply the "switch") that are in use for the duration of a call. In the more modern packet switching technology, packets of bits pass through the interconnection network. A packet is a finite series of digital bits, some of which represent control information such as the destination address and some of which represent the actual information being transmitted.

A simple way to design an interconnection network is with a cross-bar architecture. In an NxN crossbar switch, as in Figure 3.30, inputs are connected to say, horizontal wires (buses) and outputs to vertical buses. There is a switch, which can be independently closed or opened, everywhere a horizontal bus crosses a vertical bus. These are the interconnection crosspoints. An "X" in the figure indicates a closed switch, creating a path from input to output. Such a cross-bar switch is said to be a "space-division" switch in that the distinct paths through it are spatially separated.

There are a wide variety of interconnection architectures available [WU]. Some are blocking networks. These reduce the number of crosspoints at the expense of blocking. That is, not every input can be connected to a different output at the same time. An input that can't find a path to an output is said to be blocked. Blocking networks can be advantageous as they reduce the number of crosspoints, a traditional measure of switch complexity, and because it may be that traffic patterns are such that only a small number of inputs will be active at one time.

In what follows though, we will consider the case of a non-blocking packet switch, such as the crossbar, with queues at either the inputs

(Figure 3.31b) or outputs (Figure 3.31a). The results and methodology we will use were published by M.J. Karol, M.C. Hluchyj (now at Codex) and S.P. Morgan of AT&T Bell Laboratories in 1986 [KARO 86,87,88]. An alternative approach to determining the limiting throughput for input queueing, by J.Y. Hui and E. Arthurs of Bell Communications Research, was published in 1987 [HUI].

But why would one consider placing queues at the input or output of a packet crossbar switch? The answer is that queueing arises naturally as packets may arrive at a number of inputs, all destined for the same output. Only one of these packets can use an output trunk (line) at a time and the others must be queued.

In all that follows it is assumed that all packets are of the same fixed length. Time is slotted and one packet just fills one time slot. The switch is synchronous. That is, arriving packets all arrive during the same time slot (see Figure 3.32).

The switch can be designed in two ways, with the queues at the inputs or the outputs. If the switch runs at the same speed (switches) as the input and the output, then only one packet can reach a specific output during a time slot and queues must be present on the switch inputs to hold packets that must wait, as in Figure 3.31b. On the other hand, if the switch is run N times as fast as the inputs and outputs then up to N packets destined for one output can pass thru the switch at one time and the queueing will take place at the outputs, as in Figure 3.31a.

Intuitively, one would expect that queueing at inputs will result in longer average queue lengths, and thus longer average waiting times, than queueing at outputs. This is because for input queueing a packet that may be able to pass through the switch in a given time instant may be forced to wait in the input queue behind a packet whose output is busy. In what follows, this superiority of output queueing over input queueing will be quantified.

3.5.2 Output Queueing

It will be assumed that packet arrivals on the N inputs obey independent and identical Bernoulli processes. The probability that a packet arrives at each input during each time slot is p. Naturally, p is the utilization of each input. Each incoming packet has a probability of 1/N of being destined for any specific output. Successive packets on the same input are independently addressed.

Let us consider a specific output queue called the "tagged" queue. Let the random variable A be the number of arriving packets at the tagged queue during a specific time slot. A should have the binomial probabilities:

$$a_i \equiv Prob[A=i] = \binom{N}{i}(p/N)^i(1-p/N)^{N-i} \qquad i=0,1,2...N \quad (3.53)$$

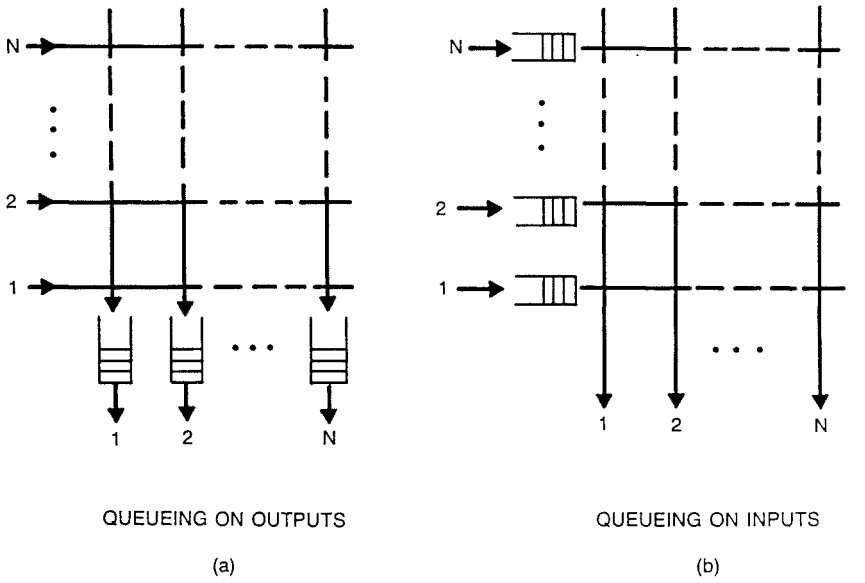

Fig. 3.31: Output and Input Queueing
Copyright 1987 IEEE

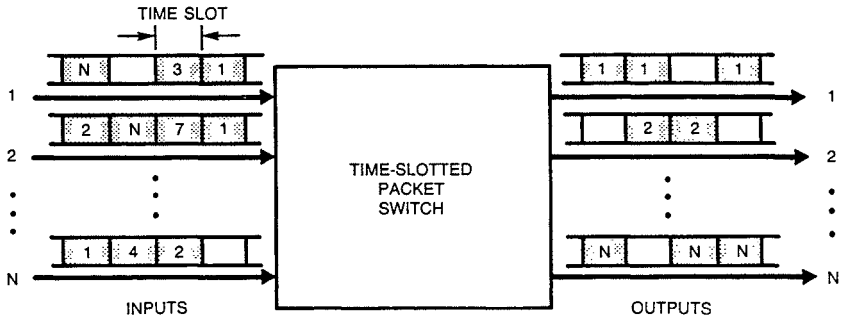

Fig. 3.32: Time-Slotted Packet Switch
Copyright 1987 IEEE

STATE TRANSITION PROBABILITIES

$$a_i \equiv \Pr(A = i) \qquad i = 0, 1, 2, \ldots$$

$$= \begin{cases} \dfrac{p^i e^{-p}}{i!} & \text{IF } N = \infty \\ \binom{N}{i}\left(\dfrac{p}{N}\right)^i \left(1 - \dfrac{p}{N}\right)^{N-i} & \text{IF } N < \infty \end{cases}$$

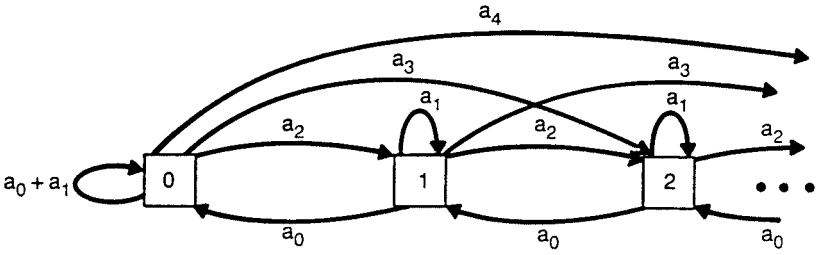

Fig. 3.33: Tagged Queue State Transition Diagram

Copyright 1987 IEEE

In the binomial distribution the term p/N is present since a packet arriving at the tagged output queue must have arrived at some input (with probability p) and then must have been destined for the tagged output (with independent probability 1/N). The moment generating function of the binomial distribution is:

$$A(z) \equiv \sum_{i=0}^{N} z^i \, Prob[A=i] = (1-\frac{p}{N}+z\frac{p}{N})^N \qquad (3.54)$$

If $N \to \infty$ the distribution of A approaches a Poisson distribution [THOM]. We have:

$$a_i \equiv Prob[A=i] = \frac{e^{-p} p^i}{i!} \qquad i=0,1,2... \qquad (3.55)$$

The moment generating function is:

$$A(z) \equiv \sum_{i=0}^{N} z^i \, Prob[A=i] = e^{-p(1-z)} \qquad (3.56)$$

Now we will write down a discrete time equation governing the tagged output queue. Let Q_m be the number of packets in the tagged queue at the end of the mth time slot. Let A_m be the number of arriving packets for the mth time slot. Then:

$$Q_m = \max(0, Q_{m-1} + A_m - 1) \qquad (3.57)$$

The -1 above is due to the packet which is transmitted by the output queue during each time slot. The number in the tagged queue is drawn in terms of a discrete time state transition diagram in Fig. 3.33. A little thought will show that it is identical to the state transition diagram of Figure 2.26 for the M/G/1 queue. Thus, using the same approach as in chapter two, one winds up with a result for the moment generating function of the distribution of the number in the queue which is remarkably similar to equation (2.248):

$$Q(z) = \frac{(1-p)(1-z)}{A(z)-z} \qquad (3.58)$$

Here, compared to (2.248), ρ is replaced by p and K(z) is replaced by A(z). There is no A(z) multiplying this expression, as K(z) does in (2.248), as the queueing system under consideration doesn't have a customer in a separate server.

Substituting A(z) for the binomial distribution from before one has:

$$Q(z) = \frac{(1-p)(1-z)}{(1-\frac{p}{N}+z\frac{p}{N})^N - z} \tag{3.59}$$

One approach now would be to invert Q(z). An alternative approach, discussed in [KARO 87] is a recursive algorithm to compute the steady state probabilities of the number in the tagged output queue. We will turn our attention though to simply calculating the average number in the queue. This can be accomplished by differentiating the above equation with respect to z and then letting $z \to 1$ (i.e. $\frac{d}{dz}\sum_{i=0}^{\infty} p_i z^i \Big|_{z=1} = \sum_{i=0}^{\infty} i p_i$ for some probability density p_i). The average number in the queue is then given by:

$$\boxed{\overline{Q} = \frac{(N-1)}{N} \times \frac{p^2}{2(1-p)} = \frac{(N-1)}{N}\,\overline{Q}_{M/D/1}} \tag{3.60}$$

Here $\overline{Q}_{M/D/1}$ is the average queue length for an M/D/1 queue modified into the context of this discrete time problem. From the above it is apparent that as $N \to \infty$, $\overline{Q} \to \overline{Q}_{M/D/1}$.

Actually, as $N \to \infty$ the equilibrium probabilities of the number in the queue also converge to those of an M/D/1 queue. To see this, one uses the previous expression for A(z) for the Poisson distribution to obtain:

$$\lim_{N\to\infty} Q(z) = \frac{(1-p)(1-z)}{e^{-p(1-z)} - z} \tag{3.61}$$

This is the moment generating function of a comparable discrete time M/D/1 queue.

Finally we will calculate the average waiting time experienced by a packet arriving at the tagged queue during the mth time slot. The service discipline is FIFO with respect to packets arriving during different time slots. However packets arriving during the same time slot are assumed to

be transmitted in random order.

To calculate the average waiting time, use will be made of Little's Law. The arrival rate into the tagged queue is the total switch arrival rate (Np) multiplied by the probability that incoming packets are destined for the tagged queue (1/N). Thus:

$$\overline{W} = \frac{\overline{Q}}{p} = \frac{(N-1)}{N} \times \frac{p}{2(1-p)} = \frac{(N-1)}{N} \overline{W}_{M/D/1} \qquad (3.62)$$

Here $\overline{W}_{M/D/1}$ is the average waiting time for a discrete time M/D/1 queue.

Figure 3.34 [KARO 86] shows the mean waiting time as a function of p for various values of N. It can be seen that the case of $N \rightarrow \infty$ provides an upper bound on the average waiting time with respect to the case of finite N.

3.5.3 Input Queueing

As for the case of output queueing, arriving packets to the N inputs of the switch follow independent and identical Bernoulli processes. The probability that a packet arrives at each input during each time slot is p. Each packet is directed to one of the N outputs with probability 1/N.

In what follows, if k packets are destined for the same output, the switch controller picks one at random (with probability 1/k) and the others wait for a new selection in the next slot. Other policies are possible [KARO 87]. One could, for example, pick a packet from the longest input queue or one can establish priorities.

In what follows, the case of input saturation is considered. This is the case when the input queue always has a packet waiting.

Let R_m^i be the number of packets at the heads of the input queues that are blocked at the end of the mth time slot and are destined for output i. That is, R_m^i is the number of packets destined for output i that are *not* selected by the switch controller during the mth time slot.

Let A_m^i be the number of packets moving to the head of the free input queues, destined for output i, during the mth time slot. An input queue is said to be *free* during the mth time slot if, and only if, a packet from it was transmitted during the (m-1)st time slot.

Again, one can write a discrete time equation:

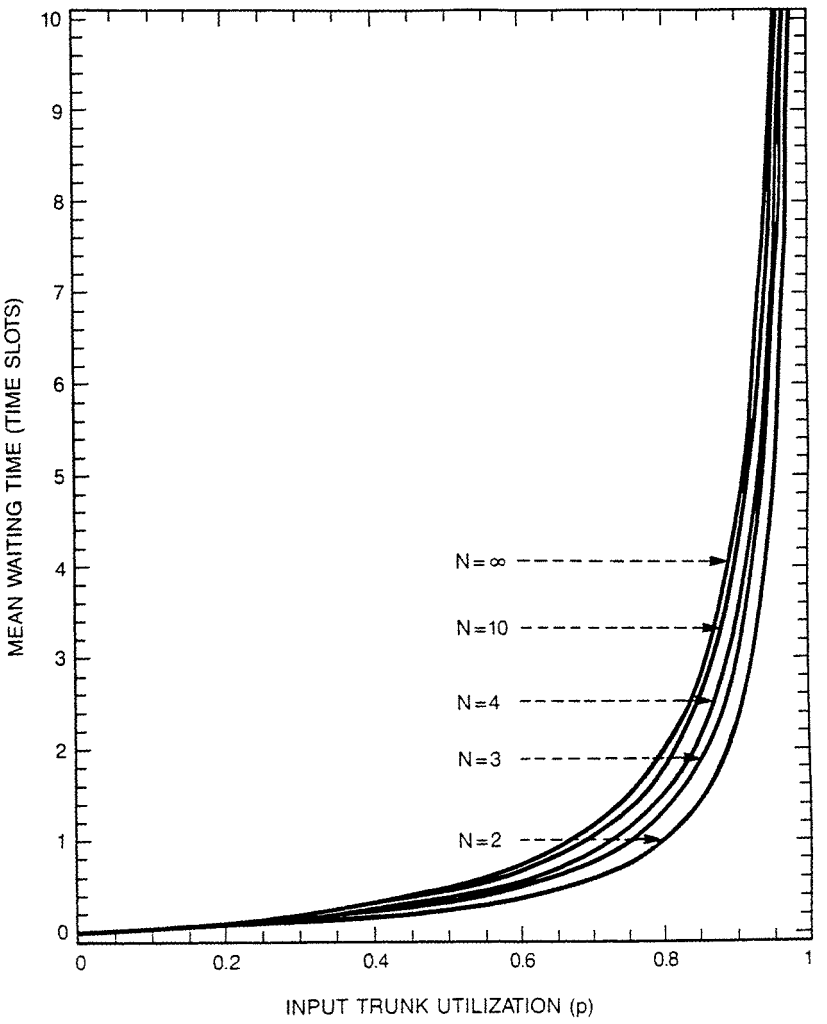

Fig. 3.34: Output Queueing Mean Waiting Time

Copyright 1987 IEEE

$$R_m^i = \max(0, R_{m-1}^i + A_m^i - 1) \qquad (3.63)$$

This equation is similar to the previous one for output queueing.

The distribution of A_m^i, the number of arriving packets to free input queues, destined for output i, during the mth time slot has the binomial probabilities

$$Prob[A_m^i = k] = \qquad (3.64)$$

$$\binom{F_{m-1}}{k}(1/N)^k(1-1/N)^{F_{m-1}-k} \qquad k=0,1,2,\ldots F_{m-1}$$

where:

$$F_{m-1} \equiv N - \sum_{i=1}^{N} R_{m-1}^i \qquad (3.65)$$

In the above binomial distribution 1/N is used instead of p/N as p=1 under input saturation. The variable F_{m-1} is the number of free input queues at the end of the $(m-1)st$ time slot, or the total number of packets passed through the switch during the (m-1)st time slot. Thus F_{m-1} is also the total number of input queues, during the mth time slot, with new packets at their heads or:

$$F_{m-1} = \sum_{i=1}^{N} A_m^i \qquad (3.66)$$

A little thought will show that it is true that $\overline{F}/N=\rho$, where ρ is the utilization of the output lines or the switch throughput. That is, $\rho=\overline{F}/N$ is the fraction of time slots on the outputs that have packets. Moreover, the equilibrium number of packets destined for output i and moving to the head of the input queues each time slot, A^i, becomes Poisson with rate ρ as $N\to\infty$. What these observations and the previous discrete time equation imply is that the output queueing M/D/1 results can be used to get an expression for the average equilibrium value of R^i. That is, as $N\to\infty$:

$$\overline{R^i} = \frac{\rho^2}{2(1-\rho)} \qquad (3.67)$$

While R_m^i does not correspond to a physical queue, all the arguments of the previous section carry through. Now an additional expression for $\overline{R^i}$ can be obtained from:

$$F_{m-1} \equiv N - \sum_{i=1}^{N} R_{m-1}^i \qquad (3.68)$$

Rearranging:

$$\frac{\sum_{i=1}^{N} R_{m-1}^i}{N} = 1 - \frac{F_{m-1}}{N} \qquad (3.69)$$

In equilibrium:

$$\overline{R^i} = 1 - \rho \qquad (3.70)$$

These two boxed expressions for $\overline{R^i}$ can only be true if as $N \to \infty$ and under saturation:

$$\rho = (2-\sqrt{2}) = .586 \qquad (3.71)$$

This is an upper bound on the throughput of any non-blocking interconnection network based switch using input queueing as $N \to \infty$. The following table illustrates the saturation throughput as a function of N (see To Look Further):

Table 3.3	
N	Saturation Throughput
1	1.0000
2	0.7500
3	0.6825
4	0.6553
5	0.6399
6	0.6302
7	0.6234
8	0.6184
∞	0.5858

This table is based on a table appearing in [BHAN]. This is confirmed by simulation results [KARO 86] plotted in Fig. 3.35.

3.6 Case Study II: Queueing on a Single-Buffered Banyan Network

3.6.1 Introduction

As has been mentioned in the previous section, there are many interconnection networks besides that of the cross-bar architecture. One popular architecture is that of the Banyan network. A 4 stage (16x16) Banyan network is illustrated in Fig. 3.36.

Rather than place queues at the inputs or outputs of the switch, buffers are placed within each of the switching elements (boxes). Each switching element is 2x2 cross-bar switch.

The Banyan network has an important self-routing property. Label the outputs with binary numbers in ascending order, from the top output to the bottom output. These are the output addresses. When a packet arrives at the input to the Banyan network, the first switching element routes the packet to its upper output if the first bit of the destination address is a 0 and to the lower output if it is a 1. The second switching element that the packet enters uses the same routing rule for the second bit in the destination address and so on... Using this routing rule a packet will find its way to the correct output no matter which input it enters. The self-routing property is important as it allows routing to be done distributively.

In what follows we will discuss the performance evaluation of a Banyan network where each queue, as in Figure 3.36, has a single buffer. This material is based on work published in 1983 [JENQ] by Y.C. Jenq, then of AT&T Bell Laboratories and now at Tektronix.

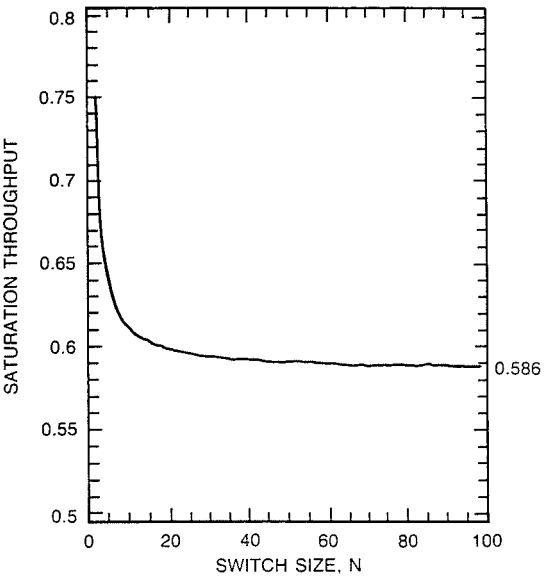

Fig. 3.35: Input Queueing Saturation Throughput
Copyright 1987 IEEE

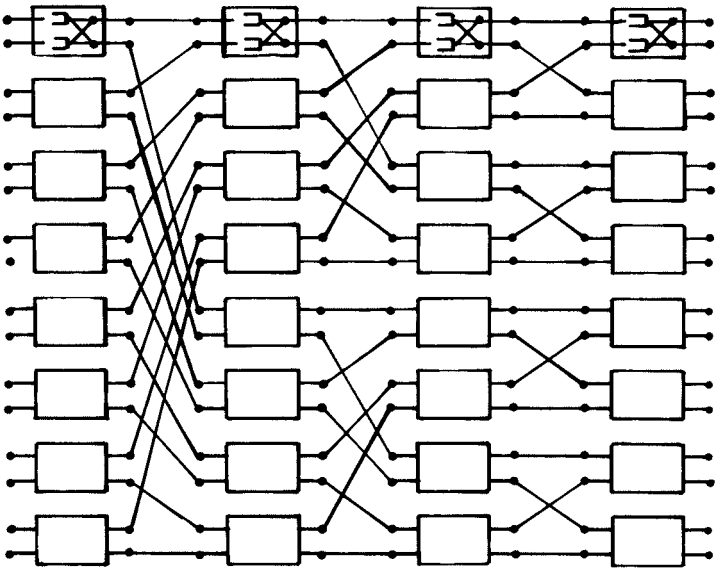

Fig. 3.36: Four Stage Banyan Network
Copyright 1983 IEEE

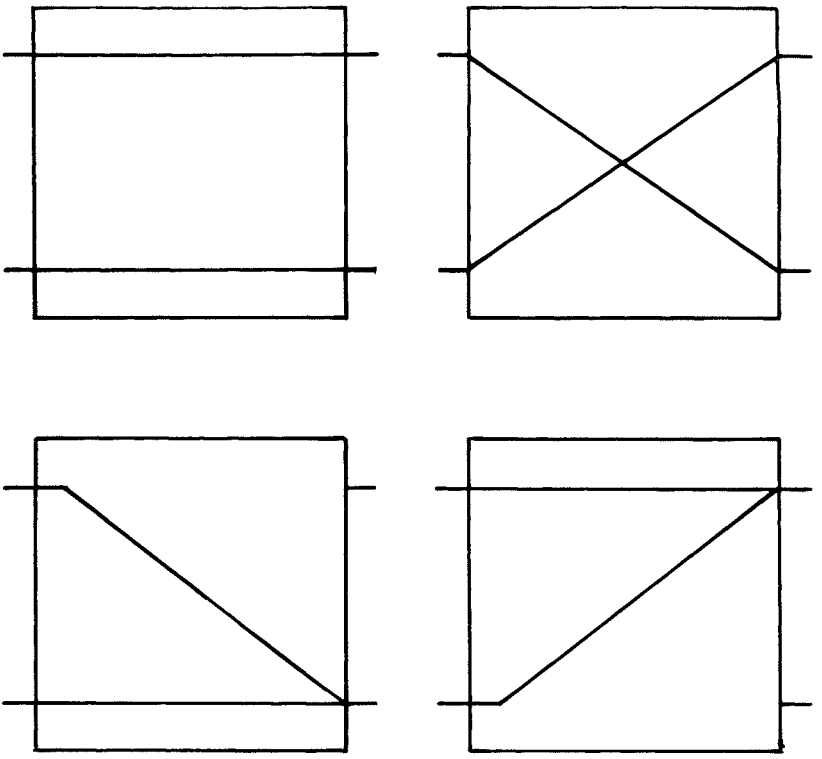

Fig. 3.37: Four Switching Possibilities for 2x2 Switch

Each 2x2 switching element can switch in fours ways. These are illustrated in Figure 3.37. If there is a packet in each buffer, then two of these switching arrangements lead to conflict and two do not. That is, conflict occurs when both packets are destined for the same output. What happens is that one packet is randomly selected to go to the next stage and the other one remains in its buffer. However a packet can only move forward if the buffer at the next stage is empty or if it is full but that packet is able to move forward. As it is put in [JENQ]: "Thus the ability for a packet to move forward depends on the state of the entire portion of the network succeeding the current stage".

In terms of implementation, this again is a synchronous system. Let a slot be τ seconds long and be composed of $\tau_1+\tau_2$ periods. During τ_1, control signals propagate from the last stage back to the first so that it is determined whether each packet can move forward or must stay in its buffer because it is blocked. During τ_2 the packets that can, move.

3.6.2 The Model Assumptions

How many states would a Markov model of the single buffered Banyan network have? Let n be the number of stages. Let each element have single buffers. Then a little thought will show that the number of states is $2^{n \cdot 2^n}$. Thus the number of states grows exponentially with the number of stages. For instance, the four stage Banyan network of Fig. 3.36 has 1.8×10^{19} states.

In order to make progress with such a model, Jenq made several simplifying assumptions.

First of all, loading is assumed to be balanced. That is, arriving packets are destined for each output with equal probability. The load on each input is $0 \leq q(1) \leq 1$.

With a balanced load the state of each switching element in stage k should be statistically the same. Thus Jenq looked at only a single switching element at stage k.

Matters are also simplified if the states of the two buffers within a switching element are assumed to be statistically independent. On one hand this assumption can be justified by noting that packets entering the inputs of a switching element originate from disjoint and independent network inputs. On the other hand, packets within the same switching element do interfere with one another. In [JENQ] models are constructed with and without this assumption and the results are shown to be similar. In what follows we will make use of this assumption.

3.6.3 The Model and Solution

Following [JENQ] closely, we now make some definitions:

$p_0(k,t)$=
the probability that a switching element buffer at stage k is empty at the beginning of the t*th* clock period.

$p_1(k,t)$=1–$p_0(k,t)$

$q(k,t)$=
the probability that a packet is ready to enter a switching element buffer at stage k during the t*th* clock period.

$r(k,t)$=
the probability that a packet in a switching element buffer at stage k is able to move (forward) into the next stage during the t*th* clock period.

The basic idea behind Jenq's performance evaluation will be to write a series of probabilistic equations, recursive in the stage number and in time, for the above quantities.

First of all we have:

$$q(k,t) = .75 \times p_1(k-1,t)p_1(k-1,t) \tag{3.72}$$

$$+ 0.5 \times p_0(k-1,t)p_1(k-1,t) + 0.5 \times p_1(k-1,t)p_0(k-1,t)$$

$$k=2,3,4...n$$

This equation relates q(k,t) to $p_{0,1}(k-1,t)$. For instance, the first term multiplies out the probability that both buffers in the k-1 stage are filled with the .75 probability that one of these two packets enters the input line of the kth stage switching element (see Figure 3.37). This term is followed by two terms corresponding to only one of the buffers in the k-1 stage switching element being filled and routing its packet, with probability 0.5, to the kth stage switching element.

Next comes an equation for r(k,t):

$$r(k,t) = [p_0(k,t)+.75p_1(k,t)] \tag{3.73}$$

$$\times[p_0(k+1,t)+p_1(k+1,t)r(k+1,t)]$$

$$k=1,2,3...n-1$$

$$r(n,t) = [p_0(n,t)+.75p_1(n,t)] \tag{3.74}$$

The first term in the brackets is the probability that, with one buffer in the kth stage switching element full, either the other buffer is empty or it full but doesn't interfere. The second term is the probability that the buffer in the k+1st stage switching element will be empty by the time the packet from the kth stage arrives.

Finally we have:

$$p_0(k,t+1) = [1-q(k,t)][p_0(k,t)+p_1(k,t)r(k,t)] \tag{3.75}$$

$$p_1(k,t+1) = 1-p_0(k,t+1) \tag{3.76}$$

The first term in the brackets is the probability that there is no incoming packet to a kth stage switching element buffer. The second term in brackets is the probability that the kth stage switching element buffer is empty or empties out.

This set of equations models the dynamics of the system. Notice that they are indexed by k, slot time. If there is an equilibrium solution these quantities should converge to time independent values for q(k), r(k), $p_0(k)$ and $p_1(k)$. The equations can, in fact, be solved iteratively for the equilibrium values.

The two performance measures of most interest, as usual, are throughput and delay. The normalized throughput, Υ, or the average number of output packets per output link per slot is:

$$\overline{\Upsilon} = p_1(k)r(k) \qquad k=1,2,3...n \tag{3.77}$$

The normalized average delay τ_{norm}, is:

$$\tau_{norm} = \frac{1}{n}\sum_{k=1}^{n}\frac{1}{r(k)} \qquad (3.78)$$

For instance, if r(k)=0.5 for some stage k, then the delay in moving through that stage is 1/.5=2 slots. The normalized average delay is computed by summing each stage's average delay and dividing by n, the number of stages. The minimal delay is thus 1.0. This corresponds to a delay of one time slot for each stage.

In Figure 3.38 the average throughput is plotted versus the independent input load parameter q(1). Below q(1)=0.4 the throughput grows linearly with input load as little blocking is occurring internal to the Banyan network. Beyond q(1)=0.4 the throughput saturates (to .45 for n=10). That throughput saturates is understandable because of the blocking that occurs when two packets in a switching element attempt to access the same output link or when a packet can not move forward.

In Figure 3.39 the normalized average delay is plotted as a function of q(1). From this graph it appears that normalized delay is in the range of 1.0 to 1.55.

3.7 Case Study III: DQDB Erasure Station Location

3.7.1 Introduction

A trend that became apparent during the 1980's was the dominant influence that high capacity fiber optic transmission systems would have on new computer network architectures. One example of this is the Distributed Queue Dual Bus (DQDB) system . DQDB is a network architecture and protocol that can serve as either a high capacity local area network or as a metropolitan area network. A metropolitan area network is a high capacity network spanning, perhaps, 50 km, that interconnects local area networks. The IEEE 802.6 standards committee is presently considering DQDB as a standard for Metropolitan Area Networks. DQDB is derived from the earlier QPSX [NEWM] which was developed at the University of Western Australia in conjunction with Telecom Australia.

DQDB consists of N stations interconnected by two fiber optic buses transmitting information in opposite directions at 150 Mbps per bus (Figure 3.40). Stations tap onto both buses and transmit information on the bus that leads to the receiving station. A clever protocol, described below, controls access to the buses.

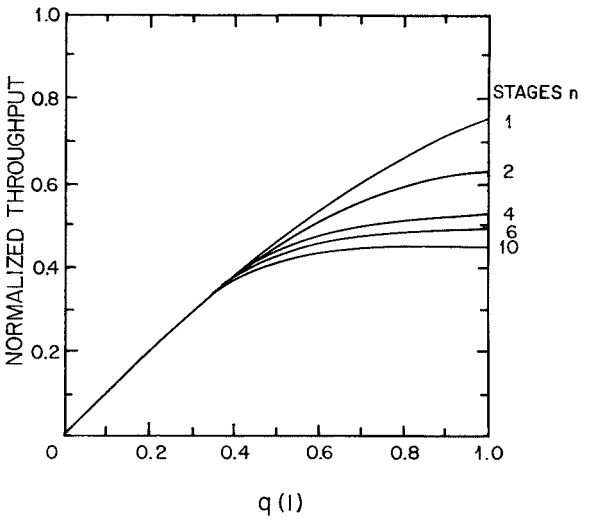

Fig. 3.38: Banyan Network Mean Throughput
Copyright 1983 IEEE

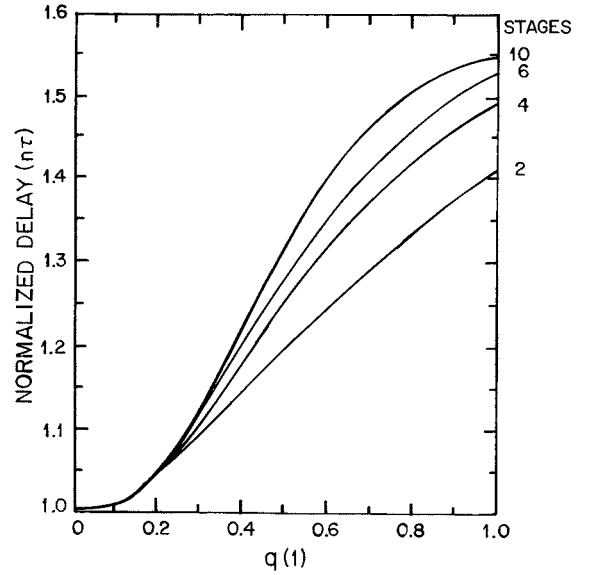

Fig. 3.39: Banyan Network Mean Delay
Copyright 1983 IEEE

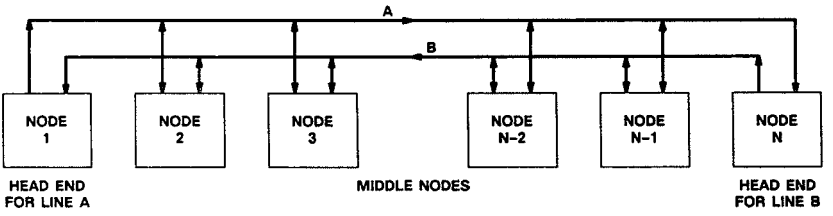

Fig. 3.40: DQDB Network Architecture

Copyright 1990 IEEE

Taps can be either passive or active. While passive taps are highly reliable, they sap light energy from the fiber and with current technology only several tens of taps are possible on a single fiber. Active taps regenerate the light signal.

In DQDB fixed length slots are generated at each head end and travel down the buses. Each slot contains a "busy bit" and a "request bit". DQDB really consists of two symmetrical transmission systems, one for each bus.

Whenever a station writes into a slot, the busy bit is set to one and prevents downstream stations from over-writing into the same slot. Whenever a station has a packet to send, it sets the first available request bit on the *opposite* bus.

DQDB is intended to implement a distributed queueing protocol. Suppose we consider information transmission on bus A. Each station counts the difference between the number of requests flowing past it, from right to left, on bus B and the number of empty (non-busy) slots flowing past it, from left to right, from upstream on bus A. When a packet is ready for transmission a station waits a number of empty slots equal to the current difference before transmitting.

If it weren't for the non-zero propagation delays encountered by the request bits, DQDB would distributively emulate a FIFO queue, network wide. That is, packets would be transmitted onto the bus(es) in the order of their arrivals. Because there is non-zero propagation delay, FIFO emulation is only an approximation and there are discrepancies in the "fairness" with which access by each station is treated [FILI].

In a purely passive implementation of DQDB, when a busy slot passes a receiver the slot can not be reused by stations further downstream. An active station may "erase" busy slots that have already reached their destinations so that stations further downstream may reuse such slots. However there is a penalty - delay is introduced by an active station as it must read the packet's address and decide whether to pass it or erase it. Therefore, a compromise has been proposed [GARR] of placing a limited number of "erasure stations" in the network that would perform this function in order to boost the network's capacity.

In what follows, the optimal locations for placing erasure stations and the associated increase in network capacity is calculated. This material is reprinted from a paper [GARR] by S-Q. Li of the University of Texas at Austin and M. Garrett of Bell Communications Research. It is reprinted with the permission of the IEEE (copyright 1990 IEEE). In 3.7.2 the network will be modeled continuously. A discrete formulation is also possible [GARR]. This is a good example of a successful attempt at tackling a distributed queueing problem.

3.7.2 Optimal Location of Erasure Nodes

by Mark W. Garrett and San-Qi Li

The first question regarding erasure nodes is: Where do they go? We simplify the problem for the moment by assuming that load is uniformly distributed among all source/destination pairs. For a dual bus network we need only analyse one side; the other behaves identically by symmetry. In one direction, the load offered by any station is proportional to the number of destination stations downstream. Therefore, the offered load is highest at the first station and decreases linearly to the last station, which sends zero because all of its traffic is sent on the opposite bus. For convenience we model the network in a continuous fashion, as if there were a station at every point on the line segment from zero to unity.

We first consider the case with no erasure nodes and no reuse of slots. Traffic is generated from any point, x, on the network according to the unidirectional transmission density function,

$$f_T(x) = 2\gamma(1-x), \qquad 0 \leq x \leq 1 \tag{3.79}$$

where γ is the maximum throughput or capacity for the single bus normalized to the case without slot reuse. The probability that a slot at point x is occupied is given by the distribution,

$$F_O(x) = \int_0^x f_T(\chi) d\chi = \gamma x(2-x) \tag{3.80}$$

For the network to be fully occupied means that all slots measured at the end of a bus are full, or, $F_O(1)=1$. This yields $\gamma=1$, i.e. without slot reuse the network capacity is the nominal value.

If we now consider destination release (DR), we have a reception density function along the unit segment,

$$f_R(x) = 2\gamma x \qquad 0 \leq x \leq 1 \tag{3.81}$$

and the occupancy distribution at any point is given by,

$$F_{O,DR}(x) = \int_0^x f_T(\chi) - f_R(\chi)\, d\chi = 2\gamma x(1-x) \qquad (3.82)$$

Setting the first derivative to zero shows that the network is now most congested in the center, rather than at the far end. The maximum throughput is found by setting $F_{O,DR}(1/2)=1$, yielding $\gamma=2$, i.e. continuous destination release results in a doubling of the network capacity.

If we now consider adding discrete erasure nodes, it is clear that traffic builds up as with no reuse, Eq. (3.80), until the first erasure node, as shown in Fig. 3.41. The traffic level is then reduced to the value of $F_{O,DR}(x)$ with continuous destination release, Eq. (3.82), and again increases monotonically until the next erasure node or the end of the network. The problem is to maximize γ by optimally locating n_e erasure nodes, subject to the constraint that the probability of occupancy be not more than 100% at all points. The local maxima in $F_O(x)$ are clearly just before each erasure node, i.e. at $x=L_k^-$, and at the end of the network. The maximum γ occurs when these maxima are just equal to unity. We now have n_e+1 constraints

$$F_O(L_k^-)=1, \quad F_O(1)=1, \quad 1\leq k\leq n_e \qquad (3.83)$$

and the same number of variables $L_1, \cdots L_{n_e}, \gamma$, where L_k denotes the location of the kth erasure node. We solve for γ and the L_k's by identifying some relationships from which we can use a simple numerical method.

From the network head end to the first erasure node, the occupancy follows Eq. (3.80). Setting $F_O(L_1^-)=1$ we have,

$$L_1^2 - 2L_1 + 1/\gamma = 0 \qquad (3.84)$$

yielding with $0\leq L_1\leq 1$,

$$L_1 = 1 - \sqrt{1-1/\gamma} \qquad (3.85)$$

Generally, just beyond any erasure node the occupancy is given by the DR expression, Eq. (3.82),

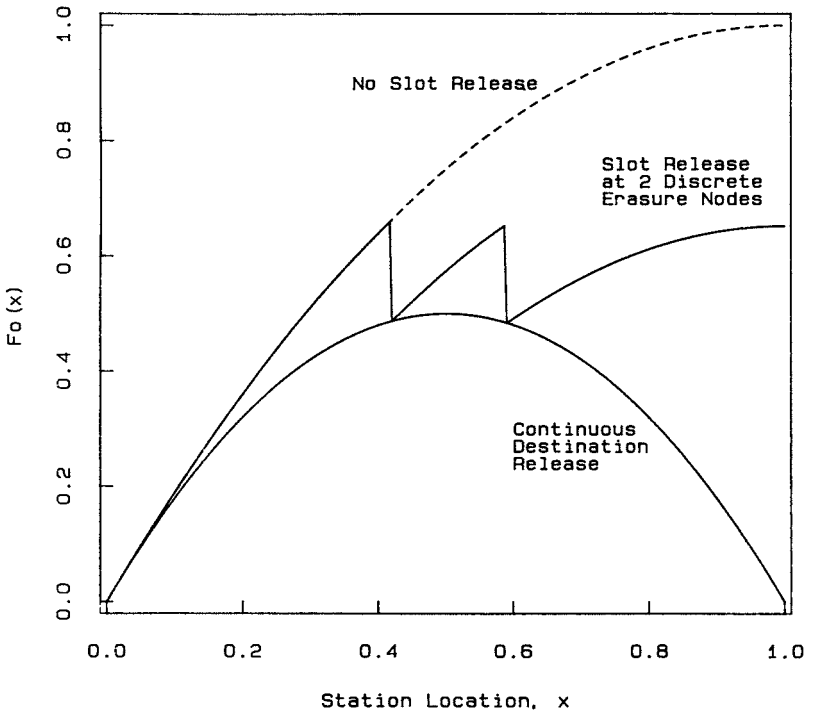

Fig. 3.41: Traffic Intensity

Copyright 1990 IEEE

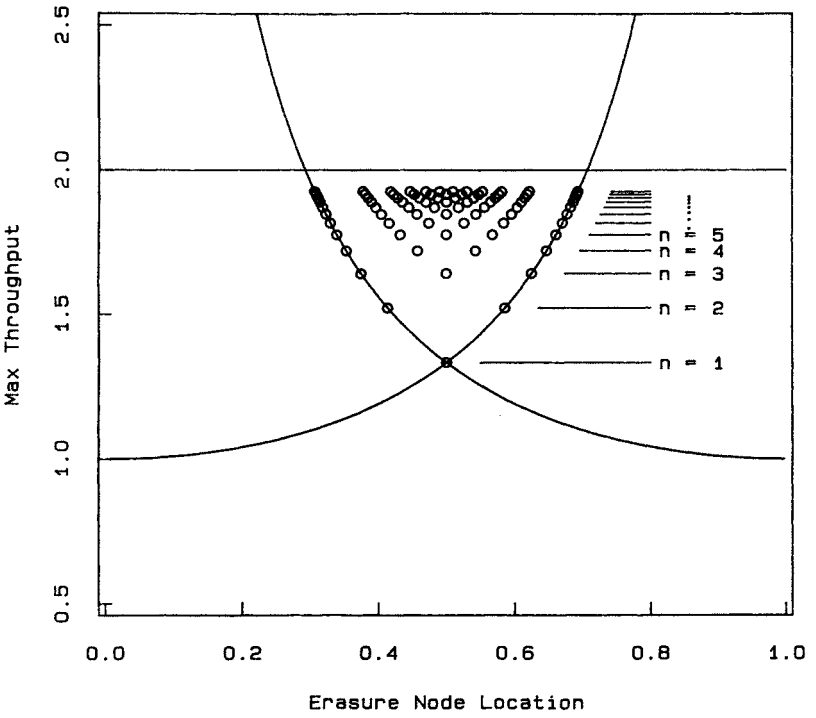

Fig. 3.42: Optimal Erasure Node Sites vs. γ

Copyright 1990 IEEE

$$F_O(L_k^+)=F_{O,DR}(L_k)=2\gamma L_k(1-L_k) \tag{3.86}$$

and between erasure nodes, using Eqs. (3.80) and (3.86),

$$F_O(x)=2\gamma L_k(1-L_k)+\int_{L_k}^{x} 2\gamma(1-\chi)d\chi$$

$$F_O(x)=-\gamma(x^2-2x+L_k^2), \quad L_k<x<L_{k+1} \tag{3.87}$$

At x=1 this gives the constraint on L_{n_e} as,

$$F_O(1) = 1 = \gamma-\gamma L_{n_e}^2 \tag{3.88}$$

yielding,

$$\boxed{L_{n_e} = \sqrt{1-1/\gamma}} \quad (3.89)$$

Equations (3.85) and (3.89) give the locations of the first and last erasure nodes as functions of γ, and are shown in Figure 3.42. For $n_e=1$, the single erasure node is optimally located at the point of intersection, x =0.5 and $\gamma=1.33$. For more than one erasure node we use Eq. (3.87) with $F_O(L_{k+1})=1$ to get,

$$1/\gamma = 2L_{k+1}-L_{k+1}^2-L_k^2, \quad 1\leq k\leq n_e \tag{3.90}$$

yielding,

$$\boxed{L_{k+1} = 1-\sqrt{1-1/\gamma-L_k^2}} \quad (3.91)$$

This set of equations, with the boundary equations (3.85) and (3.89) is easily solved numerically by searching for a value of γ for which the value of L_{n_e} given by Eq. (3.89) matches the value obtained by iterating Eq. (3.91) n-1 times, from L_1 given by Eq. (3.85). Fig. (3.42) shows the resulting locations for up to 12 erasure nodes. A substantial throughput gain is

possible for up to five or six erasure nodes, with only marginal gains thereafter [GARR].

These results show that erasure nodes are quite non-uniformly spaced. Indeed, with continuous reuse, Eq. (3.82), traffic is concentrated in the center of the network, and so erasure nodes are optimally located near the center also. We note that as $n_e \to \infty$, $L_1 \to 1-\sqrt{1/2}$. Equation (3.80) shows that this is the first location where traffic builds up to 100% occupancy when $\gamma=2$, which is the limit of γ as $n_e \to \infty$.

To Look Further

At about the time that the BCMP paper was published a number of related works also appeared. These were written by F. P. Kelly [KELL 75] [KELL 76] and by M. Reiser and H. Kobayashi [REIS 75]. The relation between these papers is discussed in [CONW 89b].

A property of certain models is "insensitivity". That is, analytical expressions for quantities such as the state probabilities may depend only on the means of relevant distributions (i.e. arrival and service distributions) rather than on the distributions themselves. An example of this involving blocking probabilities in circuit switched networks appears in [BURM]. The question of insensitivity is also examined in [SURI 83].

The consistency graph has been used by R. Lee Hamilton of The Ohio State University and E.J. Coyle of Purdue University to synthesize multihop radio networks with product form solution [HAMI 86a, 86b, 89]. This was previously not possible.

A recent generalized recursive approach to solving for the equilibrium state probabilities appears in [YANG]. This work also provides references to related work.

In [KARO 86] an interesting point is made that the same asymptotic throughput as for input queueing packet switches has also been obtained in the different context of memory interference in synchronous multiprocessor systems [BASK 76][BHAN].

Jenq's 1983 paper also examines switch delay when there are infinite sized buffers at the banyan network inputs. In [KIM] probabilistic equations are used to account for non-uniform traffic patterns in a banyan network with single or multiple buffers.

Problems

3.1 For the queueing network of Figure 3.2 solve the traffic equations for the average throughput θ_1 and θ_2.

3.2 [KOBA] Consider a state dependent open queueing network as in section 3.2.2. The service rate of the ith queue, $\mu_i(n_i)$, is a function of the number of customers in the ith queue, n_i. The arrival rate is as it is in 3.2.2 (state independent). Write out the global balance equation for state $\underline{n}$. Show that the product form solution is:

$$p(\underline{n}) = p(\underline{0}) \prod_{i=1}^{M} \frac{\theta_i^{n_i}}{\prod_{n=1}^{n_i} \mu_i(n)}$$

3.3 [KOBA] Consider a state dependent closed queueing network as in section 3.2.4. The service rate of the ith queue, $\mu_i(n_i)$, is a function of the number of customers in the ith queue, n_i. Write out the global balance equation for state $\underline{n}$. Show that the product form solution is:

$$p(\underline{n}) = \frac{1}{G(N)} \prod_{i=1}^{M} \frac{\theta_i^{n_i}}{\prod_{n=1}^{n_i} \mu_i(n)}$$

3.4 Consider a three queue Markovian cyclic network with two customers. The state transition diagram appears in Figure 3.4. By identifying and solving the appropriate local balance equations along a path back to p_{00}, solve for p_{12}.

3.5 For an open two queue Markovian series network, as in Figure 3.10, write the local balance equations involving the (n_1,n_2) state. Verify that the product form solution

$$p_{n_1 n_2} = \left(\frac{\lambda}{\mu_1}\right)^{n_1} \left(\frac{\lambda}{\mu_2}\right)^{n_2} p_{00}$$

satisfies these equations. Do the same for the global balance equations for the (n_1,n_2) state.

3.6 Consider an open network of two finite buffer Markovian queues in series (tandem). Suppose that when either queue is full the arrival and service process at the other queue will be stopped until the full queue releases a customer. Draw the state transition diagram. Verify that every transition belongs to a local balance pair and that the product form solution exists.

3.7 Consider an arbitrarily configured network of Markovian finite buffer queues with skipping. That is, when a queue is full, arriving customers skip over it. Does this network have a product form solution? Hint: Develop an argument based on state dependent service rates.

3.8 For Figure 3.14 we have:

$$\alpha_l \lambda p_a = \sum_{j=1}^{n} \beta_j \mu_l p_l = \mu_l p_l \qquad l = 1,2,3...m$$

$$\mu_l p_l = \beta_{l-m} \sum_{i=1}^{m} \mu_i p_i \qquad l = m+1, m+2, ...m+n$$

Solve for p_l, l=1,2,3...m,m+1,...m+n. Explain how you arrive at your results.

3.9 Consider the FIFO queueing system associated with the state transition diagram of Fig. 3.16. Using the consistency condition, prove that one can not have distinct service rates for the two different classes of customers, i.e. μ_1 and μ_2, and still maintain consistency. Be sure to generate the consistency graph directly from the proposed local balance equations. Note: It is true in general for FIFO networks that the service rate must not be class dependent [BASK 75].

3.10 Consider a queue where the service time is an Erlang distributed random variable consisting of N stages of independent exponential servers. Draw the state transition diagram in two dimensional form. That is, let the horizontal axis correspond to the number of customers in the queue (not including the one in service) and let the vertical axis correspond to the stage of service the customer being serviced is in. Let (0,0) correspond to an empty queueing system. Does this state transition diagram have Type A or Type B structure? Write a set of recursive equations for the equilibrium state probabilities.

3.11 Consider a queue that processes two classes of customers according to a priority discipline. That is, class 1 customers are always serviced

before class 2 customers. If a class 2 customer is being serviced and a class 1 customer arrives, the class 1 customer immediately receives service and the class 2 customer will have to start service all over again.

Draw the state transition diagram. Let the horizontal axis correspond to the number of class 1 customers in the queueing system and let vertical axis correspond to the number of class 2 customers in the queueing system. Is the structure of the state transition diagram Type A or Type B? Identify the subsets of states that can be solved for individually.

Chapter 4: Numerical Solution of Models

4.1 Introduction

The ultimate goal of much performance analysis is to generate numerical performance results for a particular system under specific conditions. These can take the form of tables or a set of performance curves. They are a variety of numerical techniques available, ranging from the brute force solution of the state equations to clever algorithms to Monte Carlo type simulation. With all these techniques the *cost* of solution in terms of solution time and computer memory requirements are important considerations. The cost of of solving a system of even moderate complexity may be prohibitively expensive for a particular technique and computer installation. There is a tradeoff between our ability to model and our ability to solve such models. There is thus an advantage to simplified models which capture the most important aspects of the system in question.

The algorithms which are available deal with closed product form networks. The algorithms generally compute performance measures for the network with N customers based on previous results for N-1 customers. One approach, due independently to J. Buzen [BUZE] and to M. Reiser and H. Kobayashi [REIS 73], is called the *convolution algorithm.* It is basically a clever recursion for the normalization constants for increasing populations of customers. What makes this useful is that a variety of performance measures can be written as functions of these normalization constants.

Another approach, due to M. Reiser and S. Lavenberg [REIS 80,81,82], is *mean value analysis.* Just as the convolution algorithm avoids the need to calculate equilibrium state probabilities, mean value analysis avoids even the calculation of the normalization constants. It is based on a number of recursions arising from fundamental concepts of queueing theory.

For models with large numbers of routing chains the *RECAL* algorithm of A. Conway and N. Georganas [CONW 86] performs the calculation with a polynomial time and space complexity. This is an improvement over the convolution and mean value analysis algorithms where the growth is exponential. Note though that the requirements of RECAL are combinatorial with respect to the number of service centers in the network. The key recursion in RECAL relates the normalization constant for the network with r closed routing chains to those of a set of networks having r-1 chains. A very different approach is the PANACEA technique of J. McKenna, D. Mitra and K.G. Ramakrishnan for large Markovian queueing

networks. Here the problem of calculating the normalization constant is transformed into a problem in integration whose solution takes the form of asymptotic power series.

Of course many practical Markovian models do not have a product form solution. Certain limited models are known to have a recursive solutions for the equilibrium state probabilities. If this is not so one can, in theory, solve the global balance equations in much the same way one solves Kirchoff's nodal equations for DC electric circuits. The difficulty is that for even for relatively simple systems the state space can be enormous. In such cases a Monte Carlo simulation may provide results of sufficient accuracy for most purposes.

In this chapter we will describe these techniques.

4.2 Closed Queueing Networks: Convolution Algorithm

4.2.1 Lost in the State Space

For a closed, product-form, queueing network we know that:

$$p(\underline{n}) = \frac{1}{G(N)} f_1(n_1) f_2(n_2) \ldots\ldots f_M(n_M) \tag{4.1}$$

where

State Independent Service Rate:

$$f_i(n_i) = \left(\frac{\theta_i}{\mu_i}\right)^{n_i} \tag{4.2}$$

State Dependent Service Rate:

$$f_i(n_i) = \frac{{\theta_i}^{n_i}}{\prod_{k=1}^{n_i} \mu_i(k)} \tag{4.3}$$

and the θ_i's are the (non-unique) solutions to the traffic equations:

$$\theta_i = \sum_{k=1}^{M} \theta_k r_{ki} \qquad i = 1,2 \cdots M \tag{4.4}$$

The difficulty in evaluating $p(\underline{n})$ lies in the determination of G(N). It does not factor out as $p(\underline{0})$ does for open networks. We will write a direct expression for it based on the fact that the sum of the probabilities of all states must be one. Use will be made of the notation from Buzen's 1973 paper [BUZE]. The legitimate state space for N customers and M queues is:

$$S(N,M) = \tag{4.5}$$

$$\left\{ \underline{n} = (n_1, n_2 \cdots n_M) \mid \sum_{i=1}^{M} n_i = N \text{ and } n_i \geq 0;\ i = 1,2 \cdots M \right\}$$

Then one can write [BRUE 80]:

$$\sum_{\underline{n} \in S(N,M)} p(\underline{n}) = 1 = \frac{1}{G(N)} \sum_{\underline{n} \in S(N,M)} \prod_{i=1}^{M} f_i(n_i) \tag{4.6}$$

or

$$\boxed{G(N) = \sum_{\underline{n} \in S(N,M)} \prod_{i=1}^{M} f_i(n_i)} \tag{4.7}$$

The constraint that the sum of the customers must be equal to N defines a hyperplane in the much larger state space of an open network. Still, the remaining state space can be enormous. Consider a single class network. There are N indistinguishable objects (customers) which must be placed in M queues. There are altogether

$$\binom{M+N-1}{N} \text{ states.}$$

If, for instance, there are ten queues and twenty-five customers this is

$$\frac{34!}{25!\ 9!} = 52{,}451{,}256 \text{ states or terms.}$$

Each term requires the multiplication of ten constants so that the total number of multiplications is over half a billion! The associated

floating point summation of terms would also require special care [ROBE 88a].

Buzen's 1973 paper showed that the normalization constant can be evaluated without recourse to the previous direct sum. This technique is called the convolution algorithm. Moreover, many important performance measures are simple functions of such normalization constants. This avoids the need to deal directly with an enormous state space. The algorithm will now be discussed.

4.2.2 Convolution Algorithm: Single Customer Class

For the case of state dependent servers we will follow the treatment in Bruell's Ph.D dissertation [BRUE 78] (later condensed into the very readable [BRUE 80]). Let g(n,m) be the normalization constant for n customers and the first m of the M queues. The reason that small letters are now used is that a series of normalization constants for increasing population sizes will be calculated up to and including G(N). By definition:

$$g(n,m) = \sum_{\underline{n} \in S(n,m)} \prod_{i=1}^{m} f_i(n_i) \tag{4.8}$$

This can be expanded as:

$$g(n,m) = \sum_{k=0}^{n} \left[\sum_{\substack{\underline{n} \in S(n,m) \\ n_m = k}} \prod_{i=1}^{m} f_i(n_i) \right] \tag{4.9}$$

Factoring one has:

$$g(n,m) = \sum_{k=0}^{n} f_m(k) \left[\sum_{\underline{n} \in S(n-k,m-1)} \prod_{i=1}^{m-1} f_i(n_i) \right] \tag{4.10}$$

This expression in brackets can be identified so:

$$\boxed{g(n,m) = \sum_{k=0}^{n} f_m(k) g(n-k,m-1)} \tag{4.11}$$

This convolution-like expression accounts for the name "convolution" algorithm. From the direct definition of g(n,m) the initial conditions for the

algorithm are:

$$g(n,1) = f_1(n) \qquad n=0,1 \cdots N$$
$$g(0,m) = 1 \qquad m=1,2 \cdots M$$

Table 4.1 illustrates the operation of the algorithm. Entries can be thought of as being calculated in this tabular form although a little thought will show that only about a column's worth or N+1 memory locations are needed ([BRUE 80]). Entries are calculated from top to bottom, left to right. Each entry is a function of the entries in the column to the left of it which are at its level or above.

The payoff from the use of this algorithm comes in the last column. A comparison of the direct expression for G(N) and g(n,m) shows that the entries of the last column, g(n,M), equals G(n). Thus the normalization constants for increasing numbers of customers are calculated as the algorithm proceeds.

For the case where the queues are not state dependent we will follow Buzen's original paper and [SCHW 87]. From before

$$g(n,m) = \sum_{\underline{n} \in S(n,m)} \prod_{i=1}^{m} f_i(n_i) \tag{4.12}$$

where

$$f_i(n_i) = \left(\frac{\theta_i}{\mu_i}\right)^{n_i} \tag{4.13}$$

This expression for g(n,m) can be decomposed as:

$$g(n,m) = \tag{4.14}$$
$$\sum_{\substack{\underline{n} \in S(n,m) \\ n_m=0}} \prod_{i=1}^{m-1} f_i(n_i) + \sum_{\substack{\underline{n} \in S(n,m) \\ n_m>0}} \prod_{i=1}^{m} f_i(n_i)$$

The first term here is just the normalization constant for the network without the mth queue or simply g(n,m-1). For the second term if one factors out $f_m(1) = \frac{\theta_m}{\mu_m}$ it can be shown that what remains is the normalization constant for m queues with n-1 customers or simply g(n-1,m). Thus:

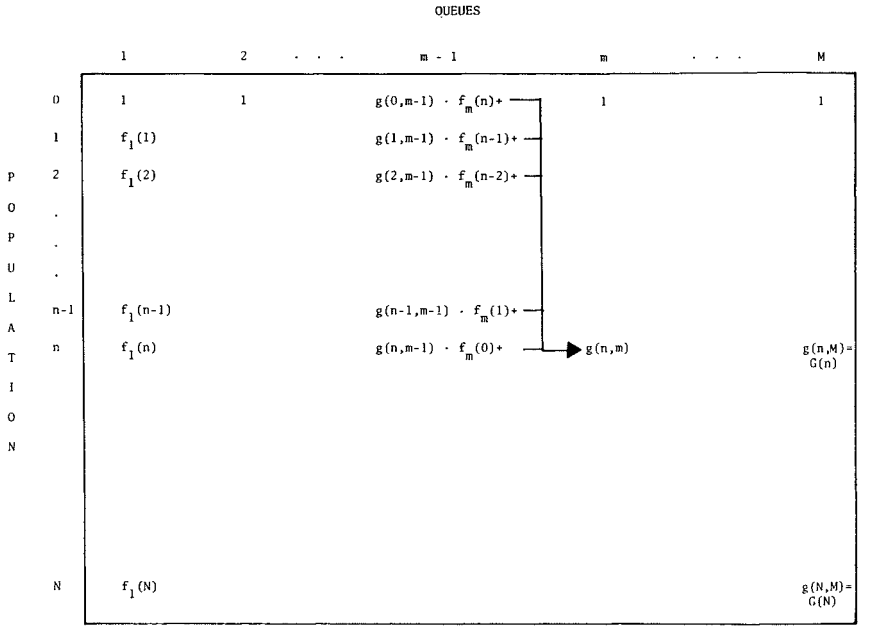
QUEUES
1
2
. . .
m - 1
m
. . .
M
POPULATION
0
1
2
n-1
n
N
1
1
g(0,m-1) · f_m(n)+
1
1
f_1(1)
g(1,m-1) · f_m(n-1)+
f_1(2)
g(2,m-1) · f_m(n-2)+
f_1(n-1)
g(n-1,m-1) · f_m(1)+
f_1(n)
g(n,m-1) · f_m(0)+
g(n,m)
g(n,M)= G(n)
f_1(N)
g(N,M)= G(N)

Table 4.1

QUEUES

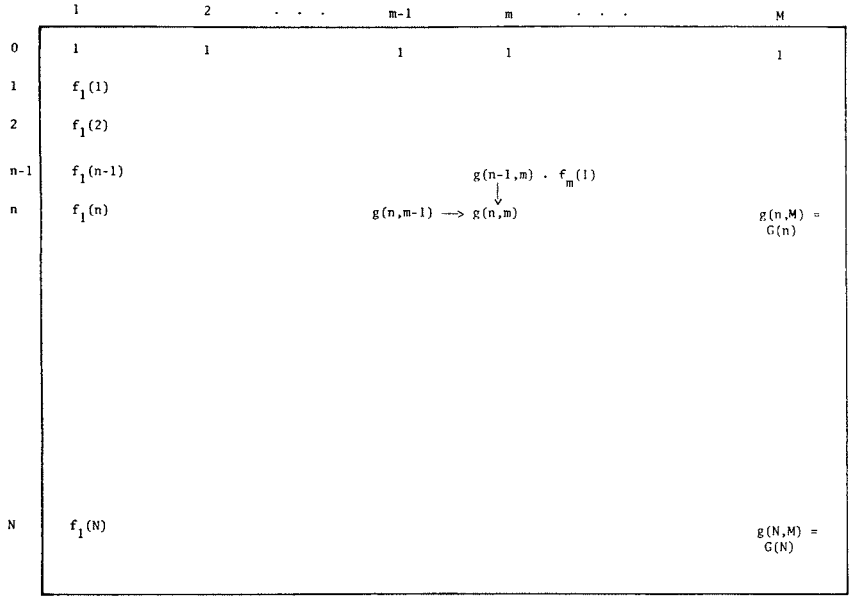

Table 4.2

$$g(n,m) = g(n,m-1) + f_m(1)g(n-1,m) \qquad (4.15)$$

This recursion is simpler than the one for the state dependent case. The computation can also be illustrated in a tabular form. From Table 4.2 it can be seen that each entry is a function of the entry to the left and the entry above. Again, only N+1 memory locations are actually needed to implement the algorithm.

What can be said of the computational complexity in time of this algorithm? It involves approximately MN additions and MN multiplications. For the state dependent recursion of Table 4.1 there are approximately $N^2/2$ additions and $N^2/2$ multiplications per column. The entire table thus requires approximately $MN^2/2$ additions and $MN^2/2$ multiplications. For the previous example of ten queues and twenty-five customers this results in 3125 multiplications rather than the half a billion mentioned before!

4.2.3 Performance Measures from Normalization Constants

Performance measures can be expressed as functions of the equilibrium state probabilities. Unfortunately this approach can lead to the same problems of excessive computation that were encountered in the calculation of the normalization constant. Fortunately, a number of important performance measures can be computed as functions of the various normalization constants which are a product of the convolution algorithm. In this section it will be shown how this can be done.

Marginal Distribution of Queue Length

Following Buzen's 1973 paper, let:

$$p(n_i = n) = \sum_{\substack{\underline{n} \in S(N,M) \\ n_i = n}} p(\underline{n}) \qquad (4.16)$$

Consider the state independent server first. To arrive at the marginal distribution it will be easier to first calculate

$$p(n_i \geq n) = \sum_{\substack{\underline{n} \in S(N,M) \\ n_i \geq n}} p(\underline{n}) \qquad (4.17)$$

or

$$p(n_i \geq n) = \sum_{\substack{\underline{n} \in S(N,M) \\ n_i \geq n}} \frac{1}{G(N)} \prod_{j=1}^{M} f_j(n_j) \tag{4.18}$$

where from before

$$f_j(n_j) = \left(\frac{\theta_j}{\mu_j}\right)^{n_j} \tag{4.19}$$

and the θ_j's satisfy the traffic equations:

$$\theta_j = \sum_{k=1}^{M} \theta_k r_{kj} \qquad j=1,2 \cdots M \tag{4.20}$$

Now every term has a factor $\left(\frac{\theta_i}{\mu_i}\right)^n$ and this can be factored out:

$$p(n_i \geq n) = \left(\frac{\theta_i}{\mu_i}\right)^n \frac{1}{G(N)} \sum_{\underline{n} \in S(N-n,M)} \prod_{j=1}^{M} f_j(n_j) \tag{4.21}$$

This summation is simply the normalization constant with N-n customers. Thus:

$$p(n_i \geq n) = \left(\frac{\theta_i}{\mu_i}\right)^n \frac{G(N-n)}{G(N)} \tag{4.22}$$

Now the key to calculating the marginal distribution is to recognize that

$$p(n_i = n) = p(n_i \geq n) - p(n_i \geq n+1) \tag{4.23}$$

so:

$$\boxed{p(n_i = n) = \left(\frac{\theta_i}{\mu_i}\right)^n \frac{1}{G(N)} \left[G(N-n) - \left(\frac{\theta_i}{\mu_i}\right) G(N-n-1)\right]} \tag{4.24}$$

Once G(1), G(2), ... G(N) are computed the marginal distributions can be found. Note that $p(n_i = n)$ depends on the non-unique θ_i. Thus the

(also non-unique) G's work in conjunction with θ_i in the previous equation to produce the actual value of $p(n_i = n)$.

Perhaps the most useful statistic that can be derived from the distribution is its mean. A little thought will show that

$$\overline{n_i}(N) = \sum_{n=1}^{N} np(n_i = n) = \sum_{n=1}^{N} p(n_i \geq n) \tag{4.25}$$

so substituting for $p(n_i \geq n)$:

$$\boxed{\overline{n_i}(N) = \sum_{n=1}^{N} \left(\frac{\theta_i}{\mu_i}\right)^n \frac{G(N-n)}{G(N)}} \tag{4.26}$$

What of the case of state dependent servers? One can start with

$$p(n_M = n) = \sum_{\substack{\underline{n} \in S(N,M) \\ n_M = n}} p(\underline{n}) \tag{4.27}$$

$$p(n_M = n) = \sum_{\substack{\underline{n} \in S(N,M) \\ n_M = n}} \frac{1}{G(N)} \prod_{j=1}^{M} f_j(n_j) \tag{4.28}$$

where from before

$$f_j(n_j) = \frac{\theta_j^{n_j}}{\prod_{n=1}^{n_j} \mu_j(n)} \tag{4.29}$$

and the θ_j's are the solutions of the traffic equations. Each term in the above summation has a term $f_M(n)$ which can be factored out. This leaves

$$p(n_M = n) = \frac{f_M(n)}{G(N)} \sum_{\substack{\underline{n} \in S(N,M) \\ n_M = n}} \prod_{\substack{j=1 \\ j \neq M}}^{M} f_j(n_j) \tag{4.30}$$

or

$$p(n_M = n) = \frac{f_M(n)}{G(N)} g(N-n, M-\{M\}) \qquad (4.31)$$

$$g(n, m-\{i\}) = \sum_{\substack{\underline{n} \in S(N,M) \\ n_i = N-n}} \prod_{\substack{j=1 \\ j \neq i}}^{m} f_j(n_j) \qquad (4.32)$$

is an auxiliary function defined in [BRUE 80]. The notation $-\{i\}$ indicates that the ith queue has been removed. The function $g(N-n, M-\{M\})$ can be calculated using the convolution algorithm where the queue of interest, M, is associated with the last column. This is possible as $g(n, M - \{M\}) = g(n, M-1)$. In other words the normalization constants that are generated after processing the first M-1 columns are the same as those generated for the network of all M queues with the Mth queue removed. If one wants the marginal distribution for a different queue, the queue must be reordered so that it is associated with the last column [BUZE 73]. Bruell and Balbo provide an alternative to this expensive procedure [BRUE 80]. Since

$$\sum_{n=0}^{N} p(n_i = n) = 1 \qquad (4.33)$$

or:

$$\sum_{n=0}^{N} \frac{f_i(n)}{G(N)} g(N-n, M-\{i\}) = 1 \qquad (4.34)$$

This can be rewritten as

$$\sum_{n=0}^{N} f_i(n) g(N-n, M-\{i\}) = G(N) \qquad (4.35)$$

or

$$g(N, M - \{i\}) = G(N) - \sum_{n=1}^{N} f_i(n) g(N-n, M-\{i\}) \qquad (4.36)$$

where the initial condition is:

$$g(0,M - \{i\}) = 1.0 = G(0)$$

This equation can be used to recursively compute $g(N,M - \{i\})$. One then has

$$p(n_i = n) = \frac{f_i(n)}{G(N)} g(N-n,M-\{i\}) \qquad (4.37)$$

where $g(n,m - \{i\})$ is defined as before. This calculation can be carried out for the ith queue without having to order the queues.

What of the expected number of customers for this state dependent case? In general it must be calculated from its definition [BRUE 80]:

$$\bar{n}_i(N) = \sum_{n=1}^{N} np(n_i = n) \qquad (4.38)$$

On the other hand, the computation of the Mth queue turns out to be similar to that of the normalization constant, G(N). One starts out with the definition of the expected number of customers

$$\bar{n}_M(N) = \sum_{n=1}^{N} np(n_M = n) \qquad (4.39)$$

and substitutes in the previous expression for the marginal probability:

$$\bar{n}_M(N) = \sum_{n=1}^{N} n \frac{f_M(n)}{G(N)} g(N-n,M-\{M\}) \qquad (4.40)$$

or since $g(n,M-\{M\}) = g(n,M-1)$ one has:

$$\bar{n}_M(N) = \frac{1}{G(N)} \sum_{n=1}^{N} nf_M(n)g(N-n,M-1) \qquad (4.41)$$

This equation is very similar to the previous convolution expression for g(n,m). Bruell and Balbo [BRUE 80] suggest performing the calculation of these two quantities together to minimize the amount of calculation.

Mean Throughput

The mean throughput for state dependent servers, is defined from first principles to be

$$\overline{\Upsilon}_i(N) = \sum_{n=1}^{N} p(n_i=n)\mu_i(n) \tag{4.42}$$

where $\mu_i(n)$ is the state dependent service rate of the ith queue. The dependence of $\Upsilon_i(N)$ on the number of customers in the system, N, is explicitly shown. Using the previous expression for $p(n_i=n)$:

$$\overline{\Upsilon}_i(N) = \sum_{n=1}^{N} \frac{f_i(n)}{G(N)} g(N-n, M-\{i\})\mu_i(n) \tag{4.43}$$

Following [BRUE 80] from the definition of $f_i(n)$ one has:

$$\overline{\Upsilon}_i(N) = \sum_{n=1}^{N} \frac{\theta_i}{\mu_i(n)} \frac{f_i(n-1)}{G(N)} g(N-n, M-\{i\})\mu_i(n) \tag{4.44}$$

Now this can simplified and rearranged to

$$\overline{\Upsilon}_i(N) = \frac{\theta_i}{G(N)} \sum_{n=1}^{N} f_i(n-1) g(N-n, M-\{i\}) \tag{4.45}$$

or with a change of variables:

$$\overline{\Upsilon}_i(N) = \frac{\theta_i}{G(N)} \sum_{n=0}^{N-1} f_i(n) g(N-n-1, M-\{i\}) \tag{4.46}$$

This last summation can be identified as being simply G(N-1) so:

$$\overline{\Upsilon}_i(N) = \theta_i \frac{G(N-1)}{G(N)} \qquad (4.47)$$

Again, the non-unique θ_i and the non-unique normalization constants combine to form the actual mean throughput.

Utilization

For a load dependent queue one can clearly write:

$$U_i(N) = 1 - p(n_i=0) \qquad (4.48)$$

However we can find $p(n_i=0)$ from the previous expression for the marginal distribution [BRUE 80]:

$$U_i(N) = 1 - \frac{g(N,M-\{i\})}{G(N)} \qquad (4.49)$$

The numerator can be computed using the previous recursive expression for the auxiliary function. If M is the last queue in the ordering this becomes simply:

$$U_M(N) = 1 - \frac{g(N,M-1)}{G(N)} \qquad (4.50)$$

A more specific expression can be found when the service rate is state independent. From [BUZE 76]

$$U_i(N) = \frac{1}{\mu_i} \times \overline{\Upsilon}_i(N) \qquad (4.51)$$

as it is evident that

$$\overline{\Upsilon}_i(N) = U_i(N) \times \mu_i \qquad (4.52)$$

Using the previous expression for throughput:

$$U_i(N) = \frac{\theta_i}{\mu_i} \frac{G(N-1)}{G(N)} \qquad (4.53)$$

Examples of the Use of the Convolution Algorithm

Example 1:

A Convolution Algorithm Example: State Independent Servers

In this example we will consider a cyclic queueing system of three queues. The service rates are $\mu_1 = \mu_2 = \mu_3 = \mu$. The mean throughput of each queue is equal and $\theta_1 = \theta_2 = \theta_3 = 1.0$ as the queues are in series. This example will be solved again, later, using the Mean Value Analysis algorithm. The table can be filled in as shown below:

Table 4.3

Loads	Stations 1	2	3
0	1.0	1.0	1.0
1	$f_1(1) = \frac{1}{\mu}$	$\frac{1}{\mu} + 1 \times \frac{1}{\mu} = \frac{2}{\mu}$	$\frac{2}{\mu} + 1 \times \frac{1}{\mu} = \frac{3}{\mu}$
2	$f_1(2) = \frac{1}{\mu^2}$	$\frac{1}{\mu^2} + \frac{2}{\mu} \times \frac{1}{\mu} = \frac{3}{\mu^2}$	$\frac{3}{\mu^2} + \frac{3}{\mu} \times \frac{1}{\mu} = \frac{6}{\mu^2}$
3	$f_1(3) = \frac{1}{\mu^3}$	$\frac{1}{\mu^3} + \frac{3}{\mu^2} \times \frac{1}{\mu} = \frac{4}{\mu^3}$	$\frac{4}{\mu^3} + \frac{6}{\mu^2} \times \frac{1}{\mu} = \frac{10}{\mu^3}$
4	$f_1(4) = \frac{1}{\mu^4}$	$\frac{1}{\mu^4} + \frac{4}{\mu^3} \times \frac{1}{\mu} = \frac{5}{\mu^4}$	$\frac{5}{\mu^4} + \frac{10}{\mu^3} \times \frac{1}{\mu} = \frac{15}{\mu^4}$
5	$f_1(5) = \frac{1}{\mu^5}$	$\frac{1}{\mu^5} + \frac{5}{\mu^4} \times \frac{1}{\mu} = \frac{6}{\mu^5}$	$\frac{6}{\mu^5} + \frac{15}{\mu^4} \times \frac{1}{\mu} = \frac{21}{\mu^5}$

From the right most column in the table we can read off

$$G(1) = \frac{3}{\mu}$$

$$G(2) = \frac{6}{\mu^2}$$

$$G(3) = \frac{10}{\mu^3}$$

$$G(4) = \frac{15}{\mu^4}$$

$$G(5) = \frac{21}{\mu^5}$$

and so we can infer that:

$$G(N) = \frac{(N+1)(N+2)}{2\mu^N}$$

Some performance measures that will also be computed later using the Mean Value Analysis algorithm will now be calculated. The mean throughput of the network as a function of the number of customers will first be calculated. The expression for this is

$$\overline{\Upsilon}_i(N) = \theta_i \frac{G(N-1)}{G(N)}$$

so

$$\overline{\Upsilon}_i(1) = \theta_i \frac{G(0)}{G(1)} = 1 \times \frac{1.0}{3/\mu} = \frac{\mu}{3}$$

$$\overline{\Upsilon}_i(2) = \theta_i \frac{G(1)}{G(2)} = 1 \times \frac{3/\mu}{6/\mu^2} = \frac{2\mu}{4}$$

$$\overline{\Upsilon}_i(3) = \theta_i \frac{G(2)}{G(3)} = 1 \times \frac{6/\mu^2}{10/\mu^3} = \frac{3\mu}{5}$$

$$\overline{\Upsilon}_i(4) = \theta_i \frac{G(3)}{G(4)} = 1 \times \frac{10/\mu^3}{15/\mu^4} = \frac{4\mu}{6}$$

$$\overline{\Upsilon}_i(5) = \theta_i \frac{G(4)}{G(5)} = 1 \times \frac{15/\mu^4}{21/\mu^5} = \frac{5\mu}{7}$$

or in general:

$$\overline{\Upsilon}_i(N) = \frac{N\mu}{N+2}$$

The mean throughput can be seen to approach a value of μ, the service rate of the queues.

The mean number of customers in each queue can be calculated using:

$$\overline{n_i}(N) = \sum_{n=1}^{N} \left(\frac{\theta_i}{\mu_i}\right)^n \frac{G(N-n)}{G(N)}$$

Thus for one customer:

$$\overline{n_i}(1) = \left(\frac{\theta_i}{\mu_i}\right) \frac{G(0)}{G(1)} = \frac{1}{\mu} \frac{1.0}{3/\mu} = .333$$

For two customers:

$$\overline{n_i}(2) = \left(\frac{\theta_i}{\mu_i}\right) \frac{G(1)}{G(2)} + \left(\frac{\theta_i}{\mu_i}\right)^2 \frac{G(0)}{G(2)}$$

$$= \frac{1}{\mu} \frac{3/\mu}{6/\mu^2} + \frac{1}{\mu^2} \frac{1.0}{6/\mu^2}$$

$$= .5 + .166 = .666$$

And finally for three customers:

$$\overline{n_i}(3) =$$

$$= \left(\frac{\theta_i}{\mu_i}\right) \frac{G(2)}{G(3)} + \left(\frac{\theta_i}{\mu_i}\right)^2 \frac{G(1)}{G(3)} + \left(\frac{\theta_i}{\mu_i}\right)^3 \frac{G(0)}{G(3)}$$

$$= \frac{1}{\mu}\frac{6/\mu^2}{10/\mu^3} + \frac{1}{\mu^2}\frac{3/\mu}{10/\mu^3} + \frac{1}{\mu^3}\frac{1.0}{10/\mu^3}$$

$$= .6 + .3 + .1 = 1.0$$

So we may deduce that:

$$\overline{n_i}(N) = .33 \times N$$

Finally, the mean waiting time spent in each queue can be calculated from Little's Law as:

$$\overline{\tau_i}(N) = \frac{\overline{n_i}(N)}{\overline{\Upsilon_i}(N)} = \frac{.33 \times N}{N\mu/N+2} = \frac{.33(N+2)}{\mu}$$

We will next do two numerical examples using the convolution algorithm. One involves state independent servers and the other involves state dependent servers. In the process we will calculate even more performance measures than in this example.

Example 2:

A Convolution Algorithm Numerical Example: State Independent Servers

Consider a cyclic queueing system of three queues. The service rates are $\mu_1 = 2.0$, $\mu_2 = 3.0$ and $\mu_3 = 6.0$. Naturally, the mean throughput of each queue is equal and $\theta_1 = \theta_2 = \theta_3 = 1.0$. The normalization constants table can be filled in as shown below:

Table 4.4

Loads	Stations 1	2	3
0	1.0	1.0	1.0
1	$f_1(1) = \frac{1}{2} = .5$	$.5 + 1 \times \frac{1}{3} =$ .833	$.833 + 1 \times \frac{1}{6} =$ 1.0
2	$f_1(2) = \frac{1}{4} = .25$	$.25 + .833 \times \frac{1}{3} =$ .527	$.527 + 1 \times \frac{1}{6} =$ .694
3	$f_1(3) = \frac{1}{8} = .125$	$.125 + .527 \times \frac{1}{3} =$ .300	$.3 + .694 \times \frac{1}{6} =$ .416
4	$f_1(4) = \frac{1}{16} = .0625$	$.0625 + .3 \times \frac{1}{3} =$ .1625	$.1625 + .416 \times \frac{1}{6} =$ .232
5	$f_1(5) = \frac{1}{32} = .03125$	$.03125 + .1625 \times \frac{1}{3} =$ .0854	$.0854 + .232 \times \frac{1}{6} =$ .124

From the right most column in the table we can read off:

$$G(1) = 1.0$$

$$G(2) = .694$$

$$G(3) = .416$$

$$G(4) = .232$$

$$G(5) = .124$$

We are now in a position to determine a number of performance measures from the normalization constants in the table. The mean throughput of the network as a function of the number of customers will first be calculated. The expression for this is

$$\overline{\Upsilon}_i(N) = \theta_i \frac{G(N-1)}{G(N)}$$

so:

$$\overline{\Upsilon}_i(1) = \theta_i \frac{G(0)}{G(1)} = 1 \times \frac{1.0}{1.0} = 1.00$$

$$\overline{\Upsilon}_i(2) = \theta_i \frac{G(1)}{G(2)} = 1 \times \frac{1.0}{.694} = 1.44$$

$$\overline{\Upsilon}_i(3) = \theta_i \frac{G(2)}{G(3)} = 1 \times \frac{.694}{.416} = 1.67$$

$$\overline{\Upsilon}_i(4) = \theta_i \frac{G(3)}{G(4)} = 1 \times \frac{.416}{.232} = 1.79$$

$$\overline{\Upsilon}_i(5) = \theta_i \frac{G(4)}{G(5)} = 1 \times \frac{.232}{.124} = 1.87$$

As the number of customers increases the throughput levels off at just under a value of two as this is the service rate of the slowest queue.

We will first calculate the marginal queue length distribution when there are four customers in the cyclic queueing network using the equation for state independent servers:

$$p(n_i = n) = \left(\frac{\theta_i}{\mu_i}\right)^n \frac{1}{G(N)}\left[G(N-n) - \left(\frac{\theta_i}{\mu_i}\right)G(N-n-1)\right]$$

Using this equation we have:

$$p(n_3 = 0) = \left(\frac{\theta_3}{\mu_3}\right)^0 \frac{1}{G(4)}\left[G(4) - \frac{\theta_3}{\mu_3} \times G(3)\right]$$

$$= \left(\frac{1}{6}\right)^0 \frac{1}{.232}\left[.232 - \frac{1}{6} \times .416\right]$$

$$= .700$$

$$p(n_3 = 1) = \left(\frac{\theta_3}{\mu_3}\right)^1 \frac{1}{G(4)}\left[G(3) - \frac{\theta_3}{\mu_3} \times G(2)\right]$$

$$= \left(\frac{1}{6}\right)^1 \frac{1}{.232}\left[.416 - \frac{1}{6} \times .694\right]$$

$$= .216$$

$$p(n_3 = 2) = \left(\frac{\theta_3}{\mu_3}\right)^2 \frac{1}{G(4)}\left[G(2) - \frac{\theta_3}{\mu_3} \times G(1)\right]$$

$$= \left(\frac{1}{6}\right)^2 \frac{1}{.232}\left[.694 - \frac{1}{6} \times 1.0\right]$$

$$= .0631$$

$$p(n_3 = 3) = \left(\frac{\theta_3}{\mu_3}\right)^3 \frac{1}{G(4)}\left[G(1) - \frac{\theta_3}{\mu_3} \times G(0)\right]$$

$$= \left(\frac{1}{6}\right)^3 \frac{1}{.232}\left[1.0 - \frac{1}{6} \times 1.0\right]$$

$$= .0166$$

$$p(n_3 = 4) = \left(\frac{\theta_3}{\mu_3}\right)^4 \frac{1}{G(4)}\left[G(0) - \frac{\theta_3}{\mu_3} \times G(-1)\right]$$

$$= \left(\frac{1}{6}\right)^4 \frac{1}{.232}\left[1.0 - \frac{1}{6} \times 0.0\right]$$

$$= .0033$$

We can also use the equation that was developed for the case of state dependent servers for this state independent server example. After all, state independent servers are a specialization of state dependent servers. We first need the auxiliary function:

$$g(N,M-\{i\}) = G(N) - \sum_{n=1}^{N} f_i(n)g(N-n,M-\{i\})$$

We will assume that the the i=3rd queue is removed from the network. The specific values, when there are four customers, are:

$$g(0,M-\{3\}) = 1.0$$

$$g(1,M-\{3\}) = G(1) - f_3(1)g(0,M-\{3\})$$

$$= 1.0 - \frac{1}{6} \times 1.0 = \frac{5}{6}$$

$$g(2,M-\{3\}) = G(2) - f_3(1)g(1,M-\{3\}) - f_3(2)g(0,M-\{3\})$$

$$= .694 - \frac{1}{6}\frac{5}{6} - \frac{1}{36}1.0 = .527$$

$$g(3,M-\{3\}) =$$

$$= G(3) - f_3(1)g(2,M-\{3\}) - f_3(2)g(1,M-\{3\}) - f_3(3)g(0,M-\{3\})$$

$$= .416 - \frac{1}{6}.527 - \frac{1}{36}\frac{5}{6} - \frac{1}{216}1.0 = .3$$

$$g(4,M-\{3\}) = G(4) - f_3(1)g(3,M-\{3\})$$

$$- f_3(2)g(2,M-\{3\}) - f_3(3)g(1,M-\{3\}) - f_3(4)g(0,M-\{3\})$$

$$= .232 - \frac{1}{6}.3 - \frac{1}{36}.527 - \frac{1}{216}\frac{5}{6} - \frac{1}{1296}1 = .1625$$

These values can be seen to match the entries in the middle column of the normalization constants table. This is because the results from the columns for stations 1 and 2 are equivalent to results for the network of three queues without the third queue. That is $g(n,M-\{M\}) = g(n,M-1)$. With these values of the normalization constants calculated we can now use the expression

$$p(n_i = n) = \frac{f_i(n)}{G(N)} g(N-n,M-\{i\})$$

to calculate the marginal state probabilities for the third queue. That is:

$$p(n_3 = 0) = \frac{f_3(0)}{G(4)} g(4,M-\{3\}) = \frac{1.0}{.232}.1625 = .700$$

$$p(n_3 = 1) = \frac{f_3(1)}{G(4)} g(3,M-\{3\}) = \frac{1/6}{.232}.3 = .216$$

$$p(n_3 = 2) = \frac{f_3(2)}{G(4)} g(2, M-\{3\}) = \frac{1/36}{.232} .527 = .0631$$

$$p(n_3 = 3) = \frac{f_3(3)}{G(4)} g(1, M-\{3\}) = \frac{1/216}{.232} \frac{5}{6} = .0166$$

$$p(n_3 = 4) = \frac{f_3(4)}{G(4)} g(0, M-\{3\}) = \frac{1/1296}{.232} 1.0 = .0033$$

A quick check will show that these marginal state probabilities do indeed sum to one.

The utilization of each queue may be calculated using:

$$U_i(N) = \frac{\theta_i}{\mu_i} \frac{G(N-1)}{G(N)}$$

Specifically for four customers:

$$U_1(4) = \frac{1}{2} \frac{.416}{.232} = .90$$

$$U_2(4) = \frac{1}{3} \frac{.416}{.232} = .60$$

$$U_3(4) = \frac{1}{6} \frac{.416}{.232} = .30$$

As one might expect, the queue with the slowest service rate has the highest utilization and the queue with the fastest service rate has the lowest utilization.

The average number of customers in each queue can be calculated using:

$$\overline{n_i}(N) = \sum_{n=1}^{N} \left(\frac{\theta_i}{\mu_i} \right)^n \frac{G(N-n)}{G(N)}$$

We will calculate this quantity for the third queue (and with four customers in the network):

$$\bar{n}_3(4) =$$

$$= \left(\frac{\theta_3}{\mu_3}\right)^1 \frac{G(3)}{G(4)} + \left(\frac{\theta_3}{\mu_3}\right)^2 \frac{G(2)}{G(4)} + \left(\frac{\theta_3}{\mu_3}\right)^3 \frac{G(1)}{G(4)} + \left(\frac{\theta_3}{\mu_3}\right)^4 \frac{G(0)}{G(4)}$$

$$= \frac{1}{6}\frac{.416}{.232} + \frac{1}{36}\frac{.694}{.232} + \frac{1}{216}\frac{1.0}{.232} + \frac{1}{1296}\frac{1.0}{.232}$$

$$= .299 + .083 + .02 + .0033$$

$$= .405$$

As a check, the average number of customers in queue three can be calculated using the marginal probability distribution that we have already calculated for the third queue:

$$\bar{n}_3(4) = 1\times.216 + 2\times.0631 + 3\times.0166 + 4\times.0033 = .405$$

The results can be seen to match.

Finally, the mean waiting time spent in the third queue can be calculated from Little's Law as:

$$\bar{\tau}_3(4) = \frac{\bar{n}_3(4)}{\Upsilon_3(4)} = \frac{.405}{1.79} = .226$$

The mean delay in the third queue consists of the mean service time of .166 seconds and the mean queueing time of .0593 seconds for a total mean waiting time of .226 seconds.

Example 3:

A Convolution Algorithm Numerical Example: State Dependent Servers

In this example we will consider a cyclic queueing system of three queues. Naturally, the mean throughput of each queue is equal and $\theta_1 = \theta_2 = \theta_3 = 1.0$. The service rates for queue 1 are:

Queue 1	
# Customers	Service Rate
1	5
2	6
3	7
4	8
5	9

The service rates for queue 2 are:

Queue 2	
# Customers	Service Rate
1	4
2	8
3	12
4	16
5	20

Queue 3 has a state independent server of rate six.

The normalization constants table can be filled in as shown below:

Table 4.5

Loads	Stations 1	2	3
0	1.0	1.0	1.0
1	$f_1(1) = \frac{1}{5} =$ .2	$.2 \times 1 + 1 \times \frac{1}{4} =$ .45	$.45 \times 1 + 1 \times \frac{1}{6} =$ .616
2	$f_1(2) = \frac{1}{5}\frac{1}{6} =$ .0333	$.0333 \times 1 +$ $.2 \times \frac{1}{4} +$ $1 \times \frac{1}{4} \times \frac{1}{8} =$ .1146	$.1146 \times 1 +$ $.45 \times \frac{1}{6} +$ $1 \times \frac{1}{6} \times \frac{1}{6} =$ .217
3	$f_1(3) = \frac{1}{5}\frac{1}{6}\frac{1}{7} =$ .00476	$.00476 \times 1 +$ $.0333 \times \frac{1}{4} +$ $.2 \times \frac{1}{8} \times \frac{1}{4} +$ $1 \times \frac{1}{12} \times \frac{1}{8} \times \frac{1}{4} =$ .0219	$.0219 \times 1 +$ $.1146 \times \frac{1}{6} +$ $.45 \times \frac{1}{6} \times \frac{1}{6} +$ $1 \times \frac{1}{6} \times \frac{1}{6} \times \frac{1}{6} =$ .0581
4	$f_1(4) =$ $\frac{1}{5}\frac{1}{6}\frac{1}{7}\frac{1}{8} =$ .000595	.00351	.0132
5	$f_1(5) =$ $\frac{1}{5}\frac{1}{6}\frac{1}{7}\frac{1}{8}\frac{1}{9} =$ .0000661	.000491	.00269

From the right most column in the table we can read off:

$$G(1) = .616$$

$$G(2) = .217$$

$$G(3) = .0581$$

$$G(4) = .0132$$

$$G(5) = .00269$$

A number of performance measures can now be determined from the normalization constants in the table. First we will calculate the mean throughput as a function of the number of customers. The expression for this, for both the state independent and state dependent cases, is

$$\overline{\Upsilon}_i(N) = \theta_i \frac{G(N-1)}{G(N)}$$

so:

$$\overline{\Upsilon}_i(1) = \theta_i \frac{G(0)}{G(1)} = 1 \times \frac{1.0}{.616} = 1.62$$

$$\overline{\Upsilon}_i(2) = \theta_i \frac{G(1)}{G(2)} = 1 \times \frac{.616}{.217} = 2.84$$

$$\overline{\Upsilon}_i(3) = \theta_i \frac{G(2)}{G(3)} = 1 \times \frac{.217}{.0581} = 3.73$$

$$\overline{\Upsilon}_i(4) = \theta_i \frac{G(3)}{G(4)} = 1 \times \frac{.0581}{.0132} = 4.40$$

$$\overline{\Upsilon}_i(5) = \theta_i \frac{G(4)}{G(5)} = 1 \times \frac{.0132}{.00269} = 4.91$$

Next we will calculate the marginal queue length distribution when there are four customers in the cyclic queueing network. We first need the auxiliary function:

$$g(N,M-\{i\}) = G(N) - \sum_{n=1}^{N} f_i(n)g(N-n,M-\{i\})$$

We will assume that the the i=3rd queue is removed from the network. The specific values, when there are four customers, are:

$$g(0,M-\{3\}) = 1.0$$

$$g(1,M-\{3\}) = G(1) - f_3(1)g(0,M-\{3\})$$

$$= .616 - \frac{1}{6}\times 1.0 = .45$$

$$g(2,M-\{3\}) = G(2) - f_3(1)g(1,M-\{3\}) - f_3(2)g(0,M-\{3\})$$

$$= .217 - \frac{1}{6}.45 - \frac{1}{36}1.0 = .114$$

$$g(3,M-\{3\}) =$$

$$= G(3) - f_3(1)g(2,M-\{3\}) - f_3(2)g(1,M-\{3\}) - f_3(3)g(0,M-\{3\})$$

$$= .0581 - \frac{1}{6}.114 - \frac{1}{36}.45 - \frac{1}{216}1.0 = .0219$$

$$g(4,M-\{3\}) = G(4) - f_3(1)g(3,M-\{3\})$$

$$- f_3(2)g(2,M-\{3\}) - f_3(3)g(1,M-\{3\}) - f_3(4)g(0,M-\{3\})$$

$$= .0132 - \frac{1}{6}.0219 - \frac{1}{36}.114 - \frac{1}{216}.45 - \frac{1}{1296}1 = .0035$$

Again, these values can be seen to match the entries in the middle column of the normalization constants table. This is because the results from the columns for stations 1 and 2 are equivalent to results for the network of three queues without the third queue. That is $g(n,M-\{M\}) = g(n,M-1)$. With these values of the normalization constants calculated we can now use the expression

$$p(n_i = n) = \frac{f_i(n)}{G(N)} g(N-n,M-\{i\}) \quad .$$

to calculate the marginal state probabilities for the third queue. That is:

$$p(n_3 = 0) = \frac{f_3(0)}{G(4)} g(4,M-\{3\}) = \frac{1.0}{.0132} .0035 = .265$$

$$p(n_3 = 1) = \frac{f_3(1)}{G(4)} g(3,M-\{3\}) = \frac{1/6}{.0132} .0219 = .277$$

$$p(n_3 = 2) = \frac{f_3(2)}{G(4)} g(2,M-\{3\}) = \frac{1/36}{.0132} .114 = .240$$

$$p(n_3 = 3) = \frac{f_3(3)}{G(4)} g(1,M-\{3\}) = \frac{1/216}{.0132} .45 = .158$$

$$p(n_3 = 4) = \frac{f_3(4)}{G(4)} g(0,M-\{3\}) = \frac{1/1296}{.0132} 1.0 = .0585$$

A quick check will show that these marginal state probabilities do indeed sum to one.

The utilization of each queue may be calculated using:

$$U_i(N) = 1 - \frac{g(N,M-\{i\})}{G(N)}$$

For the third queue and four customers this is:

$$U_3(4) = 1 - \frac{g(4,M-\{3\})}{G(4)} = 1 - \frac{.0035}{.0132} = .735$$

To calculate the utilization for the 1st or 2nd queue would require us to reorder and recompute the normalization constants table so that the desired queue would be associated with the last column.

For the expected number of customers for the Mth queue

$$\bar{n}_M(N) = \frac{1}{G(N)} \sum_{n=1}^{N} n f_M(n) g(N-n, M-1)$$

and so for four customers and the third (last) queue:

$$\bar{n}_3(4) = \frac{1}{G(4)} \sum_{n=1}^{4} n f_3(n) g(4-n, 2)$$

$$= \frac{1}{G(4)} \left[1 f_3(1) g(3,2) + 2 f_3(2) g(2,2) + 3 f_3(3) g(1,2) + 4 f_3(4) g(0,2) \right]$$

$$= \frac{1}{.0132} \left[1 \frac{1}{6} .0219 + 2 \frac{1}{36} .114 + 3 \frac{1}{216} .45 + 4 \frac{1}{1296} 1.0 \right]$$

$$= 1.46$$

As a check, the average number of customers in queue three can be calculated using the marginal probability distribution that we have already calculated for the third queue:

$$\bar{n}_3(4) = 1 \times .277 + 2 \times .240 + 3 \times .158 + 4 \times .0585 = 1.46$$

The results can be seen to match.

Finally, the mean waiting time spent in the third queue can be calculated from Little's Law as:

$$\bar{\tau}_3(4) = \frac{\bar{n}_3(4)}{\Upsilon_3(4)} = \frac{1.46}{4.40} = .332$$

The mean delay in the third queue consists of the mean service time of .166 seconds and the mean queueing time of .166 seconds for a total mean waiting time of .332 seconds.

4.3 Mean Value Analysis

Mean Value Analysis uses a number of fundamental queueing relationships to determine the mean values of throughput, delay and population size for closed product form queueing networks. It does not make use of the normalization constant, as does the convolution algorithm. MVA was developed by M. Reiser and S. Lavenberg of IBM (see [REIS 80,81,82]).

4.3.1 State (Load) Independent Servers

For simplicity let us consider a closed cyclic network consisting of M queues with N customers. The ith queue will have a service rate of μ_i. Suppose that we write an expression for $\bar{\tau}_i$, the average delay that a customer experiences at the ith queue. This has two components, the service time of that customer and the service time for all the customers before it. This is:

$$\bar{\tau}_i = \tag{4.54}$$

$$\frac{1}{\mu_i} + \frac{1}{\mu_i} \times (\textit{average number of customers present upon arrival})$$

Note that we do not worry about the residual service time of the customer in service because we are dealing with a Markovian system. What do we know about the average number of customers present upon arrival? According to the *Arrival Theorem* ([LAVE 80]) for closed exponential networks the number of customers present upon arrival has the same distribution as the equilibrium distribution for the network with one customer less. Intuitively, this follows from the Markovian nature of the network. Thus the previous equation becomes:

$$\bar{\tau}_i(N) = \frac{1}{\mu_i} + \frac{1}{\mu_i} \times \bar{n}_i(N-1) \tag{4.55}$$

where $\bar{n}_i(N)$ is the average number of customers in the ith queue when there are N customers in the network.

The second half of the MVA algorithm depends on Little's Law being applied first to the entire network of queues and then to the ith queue:

$$\bar{\Upsilon}(N) \sum_{i=1}^{M} \bar{\tau}_i(N) = N \tag{4.56}$$

$$\overline{\Upsilon}(N)\overline{\tau}_i(N) = \overline{n}_i(N) \tag{4.57}$$

Here $\overline{\Upsilon}(N)$ is the average throughput in the network with N customers. With some simple manipulation these three equations can now be listed in an algorithmic form:

MVA Algorithm (State Independent Servers)	
$\overline{n}_i(0) = 0$	$i = 1,2 \cdots M$
$\overline{\tau}_i(N) = \frac{1}{\mu_i} + \frac{1}{\mu_i} \times \overline{n}_i(N-1)$	$i = 1,2 \cdots M$
$\overline{\Upsilon}(N) = \frac{N}{\sum_{i=1}^{M} \overline{\tau}_i(N)}$	
$\overline{n}_i(N) = \overline{\Upsilon}(N)\overline{\tau}_i(N)$	$i = 1,2 \cdots M$

These algorithm equations are repeatedly solved for increasing population size.

Examples of the Use of the MVA Algorithm

Example 1: M Cyclic Queues

We will examine a very natural example that appears in [SCHW 87] which consists of an M queue cyclic network with $\mu_1 = \mu_2 = \mu_3 = \ldots\ldots = \mu_M = \mu$. This is the same example which we did before using the convolution algorithm for a three queue cyclic network. The MVA algorithm will allow us to solve for the M queue case.

The assumption that the queues are arranged in a cycle assures that the throughput of each queue is identical. The assumption that all the service rates are the same will also simplify matters since the mean number in each queue and the mean time delay through each queue will be identical.

For one customer over the queues i=1,2 .... M:

$$\overline{n}_i(0) = 0$$

$$\overline{\tau}_i(1) = \frac{1}{\mu}$$

$$\overline{\Upsilon}(1) = \frac{1}{M\frac{1}{\mu}} = \frac{\mu}{M}$$

$$\overline{n_i}(1) = \frac{\mu}{M} \times \frac{1}{\mu} = \frac{1}{M}$$

For two customers over the queues i=1,2 .... M:

$$\overline{\tau_i}(2) = \frac{1}{\mu} + \left(\frac{1}{\mu} \times \frac{1}{M}\right) = \frac{1}{\mu}\left(\frac{M+1}{M}\right)$$

$$\overline{\Upsilon}(2) = \frac{2}{M \times \frac{1}{\mu} \times \left(\frac{M+1}{M}\right)} = \frac{2\mu}{M+1}$$

$$\overline{n_i}(2) = \frac{2\mu}{M+1} \times \frac{1}{\mu} \times \frac{M+1}{M} = \frac{2}{M}$$

For three customers over the queues i=1,2 .... M:

$$\overline{\tau_i}(3) = \frac{1}{\mu} + \left(\frac{1}{\mu} \times \frac{2}{M}\right) = \frac{1}{\mu}\left(\frac{M+2}{M}\right)$$

$$\overline{\Upsilon}(3) = \frac{3}{M \times \frac{1}{\mu} \times \left(\frac{M+2}{M}\right)} = \frac{3\mu}{M+2}$$

$$\overline{n_i}(3) = \frac{3\mu}{M+2} \times \frac{1}{\mu} \times \left(\frac{M+2}{M}\right) = \frac{3}{M}$$

Finally, one can deduce a pattern in the answers so that for N customers over the queues i=1,2 .... M:

$$\overline{\tau_i}(N) = \frac{1}{\mu}\left(\frac{M+N-1}{M}\right)$$

$$\bar{\Upsilon}(N) = \frac{N\mu}{M+N-1}$$

$$\bar{n}_i(N) = \frac{N}{M}$$

These results can be seen to match those obtained previously with the convolution algorithm when M=3.

Example 2: Cyclic Queueing Network Numerical Example

We will now re-solve the numerical problem for state independent servers that we worked out using the convolution algorithm. Again, the closed cyclic queueing network consists of three queues with service rates of $\mu_1 = 2.0$, $\mu_2 = 3.0$ and $\mu_3 = 6.0$. The algorithm is initialized with:

$$\bar{n}_i(0) = 0 \qquad i=1,2,3$$

For one customer:

$$\bar{\tau}_1(1) = .500$$
$$\bar{\tau}_2(1) = .333$$
$$\bar{\tau}_3(1) = .166$$

$$\bar{\Upsilon}(1) = \frac{1}{.5 + .333 + .166} = 1.0$$

$$\bar{n}_1(1) = 1.0 \times .500 = .500$$
$$\bar{n}_2(1) = 1.0 \times .333 = .333$$
$$\bar{n}_3(1) = 1.0 \times .166 = .166$$

For two customers:

$$\bar{\tau}_1(2) = .500 + .500 \times .500 = .75$$
$$\bar{\tau}_2(2) = .333 + .333 \times .333 = .444$$
$$\bar{\tau}_3(2) = .166 + .166 \times .166 = .194$$

$$\bar{\Upsilon}(2) = \frac{2}{.75 + .444 + .194} = 1.44$$

$$\bar{n}_1(2) = 1.44 \times .75 = 1.08$$
$$\bar{n}_2(2) = 1.44 \times .444 = .639$$
$$\bar{n}_3(2) = 1.44 \times .194 = .279$$

For three customers:

$$\bar{r}_1(3) = .500 + .500 \times 1.08 = 1.04$$
$$\bar{r}_2(3) = .333 + .333 \times .639 = .546$$
$$\bar{r}_3(3) = .166 + .166 \times .279 = .212$$

$$\bar{\Upsilon}(3) = \frac{3}{1.04 + .546 + .212} = 1.67$$

$$\bar{n}_1(3) = 1.67 \times 1.04 = 1.74$$
$$\bar{n}_2(3) = 1.67 \times .546 = .912$$
$$\bar{n}_3(3) = 1.67 \times .212 = .354$$

For four customers:

$$\bar{r}_1(4) = .500 + .500 \times 1.74 = 1.37$$
$$\bar{r}_2(4) = .333 + .333 \times .912 = .637$$
$$\bar{r}_3(4) = .166 + .166 \times .354 = .225$$

$$\bar{\Upsilon}(4) = \frac{4}{1.37 + .637 + .225} = 1.79$$

$$\bar{n}_1(4) = 1.79 \times 1.37 = 2.45$$
$$\bar{n}_2(4) = 1.79 \times .637 = 1.14$$
$$\bar{n}_3(4) = 1.79 \times .225 = .40$$

The values of mean throughput can be seen to match those generated previously by the convolution algorithm. Moreover the value $\bar{n}_3(4)$ also matches.

4.4 PANACEA: Approach for Large Markovian Queueing Networks

4.4.1 Introduction

PANACEA is a software package and associated mathematical technique developed by J. McKenna (now at Bell Communications Research), D. Mitra and K. G. Ramakrishnan of AT&T Bell Laboratories that is able to solve Markovian queueing networks which are significantly larger than those which can be handled by other computational techniques. The package can solve multi-class, closed, open and mixed queueing networks. The basic idea used is to express the normalization constant as an integral which is in turn approximated by an asymptotic power series. Thus PANACEA produces approximations - though very accurate ones - along with bounds (see section 4.4.9). Like the convolution algorithm the key is to efficiently calculate the normalization constant, though by a radically different means.

The asymptotic power series used are generally power series in 1/N, where N is a generic large parameter. The number of terms required to produce a desired accuracy decreases with increasing N. Moreover, multiple classes can be handled with only incremental increases in computing time.

In what follows we will very closely follow the exposition in [McKEN 82]. It deals with the specific case of closed networks of state independent servers which are not heavily loaded. This material is among the most advanced in this book. However it is well worth reading as it represents a unique approach to the problem of queueing network calculation.

4.4.2 The Product Form Solution

There are p classes of customers and j is implicitly used for indexing class. That is, j $(1 \leq j \leq p)$ will not be explicitly written under summations or product.

First, we will consider some nomenclature. There are s service centers of the BCMP types 1,2,3,4 (see section 3.2.5). It turns out that the PANACEA approach requires a type 3 (infinite number of servers) service center in each route. This is not that overly restrictive as terminals, which are well modeled by type 3 service centers, appear throughout computer systems models. It thus becomes convenient to say that service centers 1 through q will be type 1,2 and 4 centers while service centers q+1 through s will be type 3 centers. When class and center indices appear together, the first symbol refers to class.

Next, the product form solution will be considered. The equilibrium probability of finding n_{ji} customers of class j at center i,

$1 \leq j \leq p$, $1 \leq i \leq s$, is $\pi(\underline{y}_1, \underline{y}_2, \underline{y}_3, \cdots \underline{y}_s)$ where:

$$\underline{y}_i = (n_{1i}, n_{2i}, n_{3i}, \cdots n_{pi}) \qquad 1 \leq i \leq s \tag{4.58}$$

The product form solution is then

$$\pi(\underline{y}_1, \underline{y}_2, \underline{y}_3, \cdots \underline{y}_s) = \frac{1}{G} \prod_{i=1}^{s} \pi_i(\underline{y}_i) \tag{4.59}$$

where

$$\pi(\underline{y}_i) = (\sum n_{ji})! \prod \left(\frac{\rho_{ji}^{n_{ji}}}{n_{ji}!} \right) \qquad 1 \leq i \leq q \tag{4.60}$$

$$\pi(\underline{y}_i) = \prod \left(\frac{\rho_{ji}^{n_{ji}}}{n_{ji}!} \right) \qquad q+1 \leq i \leq s \tag{4.61}$$

where

$$\rho_{ji} = \frac{\textit{expected \# of visits of class } j \textit{ customers in center } i}{\textit{service rate of class } j \textit{ customers in center } i} \tag{4.62}$$

The numerator here is equivalent to the solutions of the traffic equations, the θ_i's, discussed in chapter three. Rather than phrasing the physical meaning of the θ_i's as throughput, it is phrased here, equivalently, as the number of visits.

It follows from the normalization of probability that the normalization constant, or partition function as a physicist would call it, is

$$G(\underline{K}) = \sum_{\underline{1}' \underline{n}_1 = K_1} \cdots \sum_{\underline{1}' \underline{n}_p = K_p} \prod_{i=1}^{s} \pi_i(\underline{y}_i) \tag{4.63}$$

where K_j is the constant customer population of the jth class and $\underline{K}$ is the vector of these populations. Also $\underline{1}' \underline{n}_1$ is a vector representation of $\sum_{i=1}^{s} n_{ji}$. Expanding the above equation yields:

$$G(\underline{K})=\sum\cdots\sum\left[\prod_{i=1}^{q}\left\{(\sum n_{ji})!\prod\frac{\rho_{ji}^{n_{ji}}}{n_{ji}!}\right\}\right]\left[\prod_{i=q+1}^{s}\left\{\prod\frac{\rho_{ji}^{n_{ji}}}{n_{ji}!}\right\}\right] \tag{4.64}$$

4.4.3 Conversion to Integral Representation

The first step in the PANACEA technique is to represent the normalization constant as an expression involving integrals. Beginning with Euler's integral:

$$n! = \int_0^\infty e^{-u}\,u^n\,du \tag{4.65}$$

This representation is used to write:

$$(\sum n_{ji})! = \int_0^\infty e^{-u_i}\prod u_i^{n_{ji}}\,du_i \qquad i=1,2,3\ldots q \tag{4.66}$$

Substituting this into equation 4.64 and bringing the integrals out results in:

$$G = \int_0^\infty\cdots\int_0^\infty \exp(-\sum_{i=1}^{q}u_i)\sum_{\underline{1}'\,\underline{n}_1=K_1}\cdots\sum_{\underline{1}'\,\underline{n}_p=K_p} \tag{4.67}$$

$$\times\left[\prod_{i=1}^{q}\left\{\prod\frac{(\rho_{ji}u_i)^{n_{ji}}}{n_{ji}!}\right\}\right]\left[\prod_{i=q+1}^{s}\left\{\prod\frac{\rho_{ji}^{n_{ji}}}{n_{ji}!}\right\}\right]du_1\cdots du_q$$

We will now express G in several alternate forms. First, using the multinomial theorem:

$$G = (\prod K_j!)^{-1}\int_0^\infty\cdots\int_0^\infty \exp(-\sum_{i=1}^{q}u_i) \tag{4.68}$$

$$\times\prod\{\sum_{i=1}^{q}\rho_{ji}u_i+\sum_{i=q+1}^{s}\rho_{ji}\}^{K_j}\,du_1\cdots du_q$$

The notation may be simplified by:

$$\rho_{j0} = \sum_{i=q+1}^{s} \rho_{ji} \qquad j=1,2,3...p \tag{4.69}$$

Now the service center index i ranges only over the centers $1 \leq i \leq q$. The quantity ρ_{j0} is a weighted combination of the mean "think times" of the infinite service centers in routing the jth class. That is, the reciprocal of the service rate is the mean time a customer spends "thinking" at a terminal between the time a prompt appears and the time he or she types return.

If the routing of the jth class contains at least one infinite service (type 3) center then $\rho_{j0}>0$, otherwise $\rho_{j0}=0$. Then let the collection of indices of classes of the former case be I and let the collection of indices for the latter case be I^*. That is:

$$j \epsilon I \longleftrightarrow \rho_{j0}>0 \qquad j \epsilon I^* \longleftrightarrow \rho_{j0}=0 \tag{4.70}$$

In this notation G becomes:

$$G = [\prod_{j \epsilon I} \rho_{j0}^{K_j} / \prod_j K_j!] \int_0^\infty \cdots \int_0^\infty \exp-(\sum u_i) \tag{4.71}$$

$$\times \prod_{j \epsilon I} \left\{ 1+\sum_i \frac{\rho_{ji}}{\rho_{j0}} u_i \right\}^{K_j} \prod_{j \epsilon I^*} \left\{ \sum_i \rho_{ji} u_i \right\}^{K_j} du_1 \cdots du_q$$

In vector notation, which will be used below,

$$G = [\prod_{j \epsilon I} \rho_{j0}^{K_j} / \prod_j K_j!] \int_{Q+} e^{-\underline{1}' \underline{u}} \prod_j (\delta_{jI} + \underline{r}'_j \underline{u})^{K_j} d\underline{u} \tag{4.72}$$

where

$$\underline{u} = (u_1, u_2, u_3, \cdots u_q)'$$
$$\underline{1} = (1,1,1, \cdots 1)'$$
$$r_j = (r_{j1}, r_{j2}, \cdots r_{jq})' \qquad 1 \leq j \leq p$$
$$r_{ji} = \rho_{ji}/\rho_{j0} \text{ if } j \epsilon I$$
$$r_{ji} = \rho_{ji} \text{ if } j \epsilon I^*$$

$$\delta_{jI} = 1 \quad \text{if } j \epsilon I$$
$$\delta_{jI} = 0 \quad \text{if } j \epsilon I^*$$
$$Q^+ = \{\underline{u} \mid u_i > 0 \text{ for } all\ i\}$$

At this point in their exposition McKenna and Mitra introduce a large parameter N:

$$\beta_j \equiv K_j / N \qquad 1 \leq j \leq p \tag{4.73}$$

$$\underline{\Gamma}_j \equiv N \underline{r}_j \qquad 1 \leq j \leq p \tag{4.74}$$

Generally β_j is close to zero and the Γ_{ji} are close to one. While many choices of N are possible, McKenna and Mitra suggest:

$$N = \max_{ij} \{\frac{1}{r_{ji}}\} \tag{4.75}$$

Substituting these expressions for β_i and $\underline{\Gamma}_j$ into the last expression for G and changing variables $\underline{z} = \underline{u}/N$ results in:

$$G(\underline{K}) = [\prod_{j \epsilon I} \rho_{j0}^{K_j} / \prod_j K_j!] \int_{Q^+} e^{-\underline{1}' \underline{u}} \prod_j (\delta_{jI} + \underline{r}_j{}' \underline{u})^{K_j} d\underline{u} \tag{4.76}$$

$$\boxed{G(\underline{K}) = [N^q \prod_{j \epsilon I} \rho_{j0}^{K_j} / \prod_j K_j!] \int_{Q^+} e^{-Nf(\underline{z})} d\underline{z}} \tag{4.77}$$

where

$$f(\underline{z}) = \underline{1}' \underline{z} - \sum_{j=1}^{p} \beta_j \log(\delta_{jI} + \underline{\Gamma}_j{}' \underline{z}) \tag{4.78}$$

In all of what follows, it will be necessary to assume that the route for each class contains an infinite server center. That is:

$$\rho_{j0} > 0 \qquad j = 1,2 \ldots p \tag{4.79}$$

Thus the set I^* is empty and I contains the whole set.

4.4.4 Performance Measures

Such important performance measures as average throughput and average delay are simply related to utilization [BRUE 80]. Therefore we will concentrate on utilization. Let $u_{\sigma i}(\underline{K})$ be the utilization of the ith processor by customers of the σth class for a population distribution, by class, in the network described by $\underline{K} = (K_1, K_2, K_3, \cdots K_p)$, as before. Then, from [BRUE 80]:

$$\mu_{\sigma i}(\underline{K}) = \rho_{\sigma i} \frac{G(\underline{K} - \underline{1}_\sigma)}{G(\underline{K})} \qquad (4.80)$$

$$1 \leq \sigma \leq p \qquad 1 \leq i \leq q$$

Here $\underline{1}_\sigma$ is a vector with all components zero except for the σth component, which is one, as before. With some manipulation, see [McKEN 82], this can be rewritten as:

$$u_{\sigma i}(\underline{K} + \underline{1}_\sigma)^{-1} = \qquad (4.81)$$

$$\left\{\frac{1}{r_{\sigma i}(K_\sigma + 1)}\right\}\left[\delta_{\sigma I} + \frac{\int_{Q^+} (\underline{\Gamma}_\sigma{}' \underline{z}) e^{-Nf(\underline{z})} d\underline{z}}{\int_{Q^+} e^{-Nf(\underline{z})} d\underline{z}}\right]$$

A technical note is warranted. In a typical large network under normal (see below) operating conditions $r_{\sigma i}$ can be expected to be on the order of 1/N, because of the normalization used. Also, K_σ is on the order of N and the term in braces is either close to 0 or 1.

More generally, what has been accomplished is to represent a basic performance measure as a ratio of integrals. This alone would not lead to a computational savings. What helps further is that these integrals can be approximated quite accurately by simple asymptotic series.

4.4.5 "Normal Usage"

The asymptotic expansions that will be developed are for a network under "normal usage". Intuitively this is a network that is not too heavily loaded, say with loads under 80%. More precisely, define:

$$\underline{\alpha} \equiv \underline{1} - \sum \beta_j \underline{\Gamma}_j \tag{4.82}$$

In terms of the original network parameters:

$$\alpha_i = 1 - \sum K_j \frac{\rho_{ji}}{\rho_{j0}} \qquad i=1,2,3...q \tag{4.83}$$

Because of the multiplication of β_j and $\underline{\Gamma}_j$, α is independent of N. Physically, α_i is a measure of the unutilized processing capability of the ith service center. Positive α_i corresponds to less than "normal" processor utilizations and negative values to very high utilizations. In what follows it is assumed that $\alpha_i > 0$, $i=1,2,3...q$. This is normal usage. Qualitatively different expansions are needed when some of the α_i are negative (very high utilization).

We close this section by noting that it is shown in [McKEN 82] that, asymptotic with network size, and for all $\alpha_i > 0$:

$$u_i = \textit{utilization of ith processor} \approx 1-\alpha_i \tag{4.84}$$

This expression directly relates the parameters α_i's to the actual utilization.

4.4.6 Some Transformations

We will now look at a transformation of the previous integral, one of whose purposes is to bring the expression for utilization into a form that is amenable to an asymptotic expression. We start by adding and subtracting terms in the exponent:

$$\int_{Q^+} e^{-Nf(\underline{z})} d\underline{z} = \int_{Q^+} e^{-Nf(\underline{z})+N\sum\beta_j(\underline{\Gamma}_j' \underline{z})-N\sum\beta_j(\underline{\Gamma}_j' \underline{z})} d\underline{z} \tag{4.85}$$

$$\int_{Q^+} e^{-Nf(\underline{z})} d\underline{z} = \tag{4.86}$$

$$\int_{Q^+} e^{-N\underline{\alpha}' \underline{z}} \exp-[N\sum_j \beta_j \{\underline{\Gamma}_j' \underline{z} - \log(1+\underline{\Gamma}_j' \underline{z})\}] d\underline{z}$$

$$\int_{Q^+} e^{-Nf(\underline{z})} d\underline{z} = \tag{4.87}$$

$$N^{-q} \int_{Q^+} e^{-\underline{\alpha}' \underline{u}} \exp-[\sum_j \beta_j \{\underline{\Gamma}_j' \underline{u} - N \log(1+\frac{1}{N}\underline{\Gamma}_j' \underline{u})\}] d\underline{u}$$

Here $\underline{u}=N\underline{z}$. Next let's make a change of variables:

$$v_i \equiv \alpha_i \mu_i \qquad 1 \leq i \leq q \tag{4.88}$$

Let's also normalize Γ_{ji}:

$$\tilde{\Gamma}_{ji} \equiv \Gamma_{ji}/\alpha_i \tag{4.89}$$

Note that:

$$\underline{\Gamma}_j' \underline{u} \equiv \underline{\tilde{\Gamma}}_j' \underline{v} \tag{4.90}$$

From the immediately preceding integral:

$$\int_{Q^+} e^{-Nf(\underline{z})} d\underline{z} = \frac{N^{-q}}{\prod \alpha_i} \int_{Q^+} e^{-\underline{1}' \underline{v}} H(N^{-1}, \underline{v}) d\underline{v} \tag{4.91}$$

where:

$$H(N^{-1}, \underline{v}) \equiv e^{s(N^{-1}, \underline{v})} \tag{4.92}$$

$$s(N^{-1}, \underline{v}) \equiv -\sum_{j=1}^{p} \beta_j \{\underline{\tilde{\Gamma}}_j' \underline{v} - N \log(1+\frac{1}{N}\underline{\tilde{\Gamma}}_j' \underline{v})\} \tag{4.93}$$

The other reason for making this transformation is that $H(N^{-1},\underline{v})$ will make it possible, later, to determine error bounds for the asymptotic series.

Repeating the transformation for the integral $\int (\underline{\Gamma}_\sigma' \underline{z}) e^{-Nf(\underline{z})} d\underline{z}$ results in a final expression for the utilization

$$u_{\sigma i} (\underline{K}+\underline{1}_\sigma)^{-1} = \left\{ \frac{\rho_{\sigma 0}}{\rho_{\sigma i}(K_\sigma+1)} \right\} \tag{4.94}$$

$$\times \left[1+\frac{1}{N} \frac{\int_{Q^+} e^{-\underline{1}' \underline{v}} (\tilde{\underline{\Gamma}}_{\sigma}' \underline{v}) H(N^{-1}, \underline{v}) d\underline{v}}{\int_{Q^+} e^{-\underline{1}' \underline{v}} H(N^{-1}, \underline{v}) d\underline{v}} \right]$$

or more compactly

$$u_{\sigma i}(\underline{K}+\underline{1}_{\sigma})^{-1} = \left\{ \frac{\rho_{\sigma 0}}{\rho_{\sigma i}(K_{\sigma}+1)} \right\} \left[1+\frac{1}{N} \frac{I_{\sigma}^{(1)}(N)}{I(N)} \right] \qquad (4.95)$$

where $I_{\sigma}^{(1)}(N)$ and $I(N)$ are the integrals of the numerator and denominator, respectively.

4.4.7 Asymptotic Expansions

The PANACEA procedure is to first obtain the power series

$$H(N^{-1}, \underline{v}) = \sum_{k=0}^{\infty} \frac{h_k(\underline{v})}{N^k} \qquad (4.96)$$

and integrate to obtain:

$$A_k \equiv \int_{Q^+} e^{-\underline{1}' \underline{v}} h_k(\underline{v}) d\underline{v} \qquad (4.97)$$

These A_k can be used so that the previous integral is:

$$I(N) \approx \sum_{k=0}^{\infty} \frac{A_k}{N^k} \qquad (4.98)$$

Now

$$h_k(\underline{v}) = \frac{1}{k!}\frac{\partial^k}{\partial(1/N)^k}H(0,\underline{v}) \qquad k=0,1,2... \tag{4.99}$$

and recall that:

$$H(N^{-1},\underline{v}) = e^{s(N^{-1},\underline{v})} \tag{4.100}$$

From the earlier definition of s it can be noted that for fixed $\underline{v}\epsilon Q^+$, $s(N^{-1},\underline{v})$ and thus $H(N^{-1},\underline{v})$ are functions of N^{-1} and are analytic in $Re(N^{-1})>\epsilon(\underline{v})$ where $\epsilon(\underline{v})<0$.

We can write

$$s^{(k)}(0,\underline{v}) = -k!f_{k+1}(\underline{v}) \qquad k=1,2,3... \tag{4.101}$$

where

$$f_k(\underline{v}) = \frac{(-1)^k}{k}\sum_j \beta_j(\tilde{\underline{\Gamma}}_j'\,\underline{v})^k \qquad k=1,2,3... \tag{4.102}$$

The derivatives of H can be expressed as:

$$H^{(k+1)}(N^{-1},\underline{v}) = \tag{4.103}$$
$$\sum_{m=0}^{k}\binom{k}{m}s^{(k+1-m)}(N^{-1},\underline{v})H^{(m)}(N^{-1},\underline{v}) \qquad k=0,1,2...$$

From the above the $\{h_k(\underline{v})\}$ can be generated recursively:

$$h_0(\underline{v}) = 1 \tag{4.104}$$

$$h_{k+1}(\underline{v}) = -\frac{1}{k+1}\sum_{m=0}^{k}(k+1-m)f_{k+2-m}(\underline{v})h_m(\underline{v}) \qquad k=0,1,2...$$

Specifically:

$$h_0(\underline{v}) \equiv 1 \tag{4.105}$$

$$h_1(\underline{v}) = -f_2(\underline{v})$$

$$h_2(\underline{v}) = -f_3(\underline{v}) + \frac{1}{2} f_2^2(\underline{v})$$

$$h_3(\underline{v}) = -f_4(\underline{v}) + f_2(\underline{v}) f_3(\underline{v}) - \frac{1}{6} f_2^3(\underline{v})$$

To summarize

$$I(N) \approx \sum_{k=0}^{\infty} \frac{A_k}{N^k} \tag{4.106}$$

where the $\int e^{-\underline{1}' \underline{v}} h_k(\underline{v}) d\underline{v}$ and the $\{h_k(\underline{v})\}$ are obtained recursively from the above equation. The calculation of $I_\sigma^{(1)}(N)$ is quite similar [McKEN 82]. But the calculation of the A_k's can be put into a quite interesting context:

4.4.8 The Pseudonetworks

It turns out that the compositions of the coefficients $\{A_k\}$ and $\{A_{\sigma,k}^{(1)}\}$ are related to the normalization constant of a certain network called the pseudonetwork. To compute the first few elements of $\{A_k\}$ and $\{A_{\sigma,k}^{(1)}\}$ it is only necessary to consider the pseudonetwork with small populations. Thus one can use an efficient algorithm for small populations, such as the convolution algorithm, in the process of calculating the coefficients of the asymptotic series.

For example:

$$A_3 = \int_{Q^+} e^{-\underline{1}' \underline{v}} h_3(\underline{v}) d\underline{v} \tag{4.107}$$

$$A_3 = \int_{Q^+} e^{-\underline{1}' \underline{v}} \{-f_4(\underline{v}) + f_2(\underline{v}) f_3(\underline{v}) - \frac{1}{6} f_2^3(\underline{v})\} d\underline{v} \tag{4.108}$$

Let's call the third term A_{33}. Using the earlier expansion for $f_2(\underline{v})$:

$$A_{33} = -\frac{1}{48} \sum_j \beta_j^3 \int e^{-\underline{1}' \underline{v}} (\tilde{\Gamma}_j' \underline{v})^6 d\underline{v} \tag{4.109}$$

$$= -\frac{1}{16} \sum_{j \neq k} \beta_j^2 \beta_k \int e^{-\underline{1}' \underline{v}} (\tilde{\Gamma}_j' \underline{v})^4 (\tilde{\Gamma}_k' \underline{v})^2 d\underline{v}$$

$$-\frac{1}{48}\sum_{j\neq k\neq l}\beta_j\beta_k\beta_l\int e^{-\underline{1}'\underline{v}}(\tilde{\underline{\Gamma}}_j'\,\underline{v})^2(\tilde{\underline{\Gamma}}_k'\,\underline{v})^2(\tilde{\underline{\Gamma}}_l'\,\underline{v})^2\,d\underline{v}$$

The subscripts range over [1,p]. The typical integral in the composition of the asymptotic expansion coefficients is, within a multiplicative constant:

$$g(\underline{m}) = g(m_1,m_2,m_3,\cdots m_p) \equiv \tag{4.110}$$

$$\frac{1}{(\prod_j m_j!)}\int_{Q^+} e^{-\underline{1}'\underline{u}}\prod_j(\tilde{\underline{\Gamma}}_j'\,\underline{u})^{m_j}\,d\underline{u}$$

This should be familiar since if we take equation 4.76 for the partition function, assume no infinite service centers (I is empty) it becomes the similar:

$$G(\underline{K}) = \frac{1}{(\prod_j K_j!)}\int_{Q^+} e^{-\underline{1}'\underline{u}}\prod_j(\underline{r}_j'\,\underline{u})^{K_j}\,d\underline{u} \tag{4.111}$$

Because of this similarity, $g(\underline{m})$ is the normalization constant of a certain network, the "pseudonetwork". The pseudonetwork is closed and has no infinite service centers. As in the original network there are exactly q processing centers and p classes of customers. The service rate of jth class customers in the ith service center is $\tilde{\Gamma}_{ji}$. The population distribution by class in vector form is $(m_1,m_2,m_3,\cdots m_p)$.

Thus we can generate the $\{A_k\}$ as functions of the normalization constants of the pseudonetwork:

$$A_0 = 1 \tag{4.112}$$

$$A_1 = -\sum_j\beta_j\,g(2\cdot\underline{1}_j)$$

$$A_2 = 2\sum_j\beta_j\,g(3\cdot\underline{1}_j)+3\sum_j\beta_j^2\,g(4\cdot\underline{1}_j)+\frac{1}{2}\sum_{j\neq k}\beta_j\beta_k\,g(2\cdot\underline{1}_j+2\cdot\underline{1}_k)$$

$$A_3 = -6\sum_j\beta_j\,g(4\cdot\underline{1}_j)-20\sum_j\beta_j^2\,g(5\cdot\underline{1}_j)-15\sum_j\beta_j^3\,g(6\cdot\underline{1}_j)$$

$$-2\sum_{j\neq k}\beta_j\beta_k\, g(2\cdot\underline{1}_j+3\cdot\underline{1}_k)-3\sum_{j\neq k}\beta_j^2\beta_k\, g(4\cdot\underline{1}_j+2\cdot\underline{1}_k)$$

$$-\frac{1}{6}\sum_{j\neq k\neq l}\beta_j\beta_k\beta_l\, g(2\cdot\underline{1}_j+2\cdot\underline{1}_k+2\cdot\underline{1}_l)$$

Here j,k,l are class indices in the range [1,p].

It has been found that the above four terms yield a reasonable approximation to I(N). In the above at most three classes have a non-zero number of customers and the total population is at most 6 (for the similar $\{A_{\sigma,k}^{(1)}\}$ it is 7, see [McKEN 82]). The small population sizes involved make the use of the convolution algorithm practical.

4.4.9 Error Analysis

In a practical sense, it is important to know how accurate the PANACEA approximation is. It turns out that the errors in estimating the integral from the use of m terms is of the same order as the (m+1)th term as $N\to\infty$.

That is [McKEN 82]:

$$\frac{A_m}{N^m} < I(N)-\sum_{k=0}^{m-1}\frac{A_k}{N^k} < 0 \qquad m=1,3,5... \qquad (4.113)$$

$$0 < I(N)-\sum_{k=0}^{m-1}\frac{A_k}{N^k} < \frac{A_m}{N^m} \qquad m=2,4,6...$$

This is a consequence of H being completely monotonic with derivatives alternating in sign.

We end here by noting that the convolution algorithm actually is used in the PANACEA package to calculate the normalization constants of the pseudonetworks. Under normal usage it is found that only four leading coefficients give high accuracy estimates.

4.5 Discrete Time Queueing Systems

In all the queueing systems that have been discussed so far, time has been assumed to be continuous. That is, arrivals and departures may take place at any instant of time. Certain practical systems operate in a

different mode involving discrete time. Here events may only be allowed to occur at periodic, equi-spaced instants.

As an example, consider two stations (Figure 4.1a) connected to a common communication medium such as a coaxial cable or radio channel. Assume that time is "slotted", being comprised of equi-width slots (Figure 4.1b). Packet length, slot width and channel speed are such that one packet can be transmitted by one station on the bus during one slot.

A packet will arrive at a station during a slot with probability a. For the purposes of this example we will assume that each station can hold only a single packet. Once a packet is in a station, further arrivals are blocked.

If a station has a packet it transmits it during each slot with probability s. A packet can only be transmitted successfully during a slot by one of the stations if the other station does not transmit a packet at the same time. If packets are transmitted by both stations during the same slot they overlap ("collide") and become garbled so that neither is successfully transmitted. Both must be retransmitted later.

Even though the protocol being described results in occasional collisions, it has the virtue of being decentralized and straightforward to implement.

We will take a numerical approach that involves calculating the state probabilities of the system and then calculating performance measures from these state probabilities. As an example, the state transition diagram of this system is shown in Figure 4.2. There are clearly four states and a number of transitions between them.

This state transition diagram is different from the ones that have been seen before as time is now discrete. The values associated with the transitions are not rates but the probabilities of making each transition during each slot. Put another way, each slot finds the system in some state. At the end of the slot the system must transit one of the transitions leaving the state so that it can enter a state (perhaps the same one) for the next slot. Some transition must be chosen and the sum of the probabilities of all the outgoing transitions must equal one.

Note that whereas in a Markovian continuous time queueing system the time spent in a state is an exponential random variable, for this discrete time system it is a geometrically distributed random variable.

Let's examine some of the transitions in Figure 4.2. If both buffers are empty, each may receive a packet with probability a^2 to push the system into state 3. If during the next slot there is either a collision (with probability s^2) or no attempted transmissions (with probability $(1-s)^2$), the system will remain in the same state. Perhaps then station B will successfully transmit its packet with probability s(1-s) followed by a successful transmission by station A with probability s(1-a).

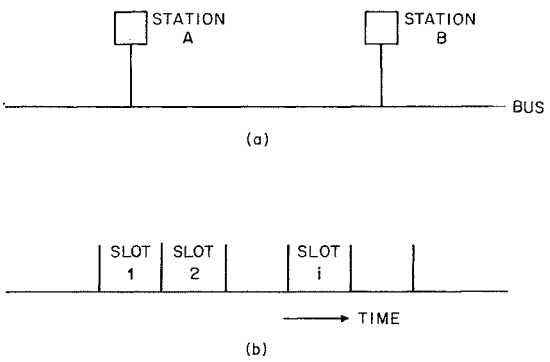

Fig. 4.1: Two Station Slotted Network

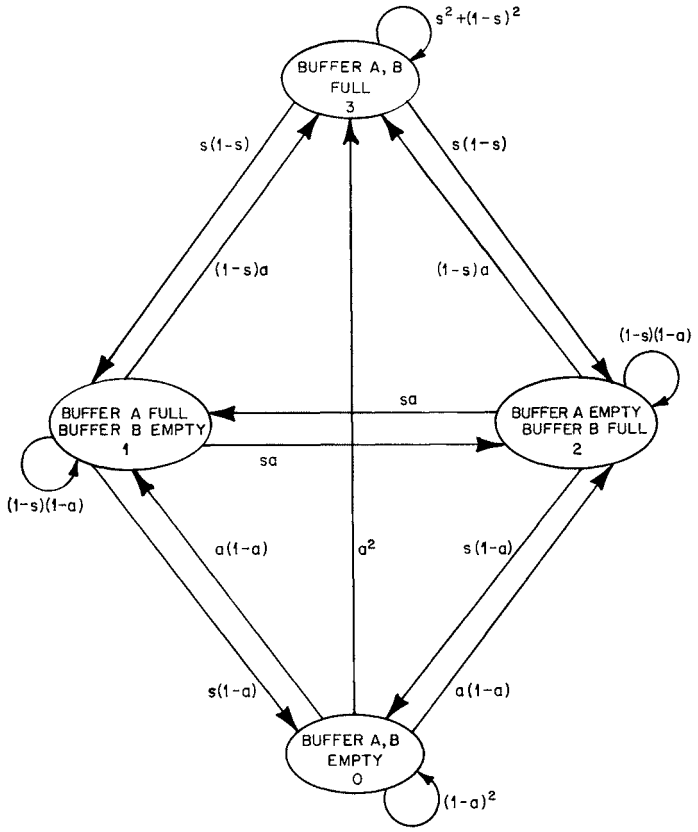

Fig. 4.2: Two Station State Transition Diagram

We can write balance equations for such discrete time Markov chains as

$$\underline{\pi}_j = \sum_i \underline{\pi}_i p_{ij} \tag{4.114}$$

$$\sum_i \pi_i = 1 \tag{4.115}$$

where p_{ij} is the probability of going from state i to j and π_j is the equilibrium probability of state j. In matrix form this can be written as

$$\boxed{\underline{\pi} = \underline{\pi P} \qquad (4.116)}$$

where $\underline{\pi} = [\pi_0\ \pi_1\ \pi_2 \cdots]$ and $\underline{P}$ is the transition matrix. That is, the ijth entry of $\underline{P}$ is p_{ij}.

A different and computationally prefarable solution [WIES] solution approach goes as follows. Let $\underline{\pi}^{(0)}$ be the state probabilities at slot 0. Then the state probabilities can be generated at each slot through the recursion:

$$\boxed{\underline{\pi}^{(n)} = \underline{\pi}^{(n-1)}\underline{P} \qquad (4.117)}$$

If the recursion is run long enough it will converge to the equilibrium state probabilities. A test for convergence such as

$$|\pi_i(n) - \pi_i(n-1)| \leq 10^{-5} \quad \text{for } each\ i$$

can be used [WIES]. Work by J.E. Wieselthier at the U.S. Naval Research Laboratory and by A. Ephremides at the University of Maryland [WIES] on discrete time models of multiple access protocols has used this solution method for state spaces of up to 700 states. It was found that a renormalization of the probabilities should be performed each iteration to prevent round-off errors from becoming a problem.

As is often the case, performance measures can be calculated for the simple model of Figure 4.2 as functions of the state probabilities. As an example, a little thought will show that the throughput of the channel is:

$$Throughput = \pi_1[s] + \pi_2[s] + \pi_3[2s(1-s)] \qquad (4.118)$$

The following section on the simulation of communication networks is authored by James F. Kurose of the University of Massachusetts, Amherst, and by Hussein T. Mouftah of Queen's University, Kingston, Ontario. It originally appeared as part of [KURO], copyright 1988 IEEE, and is reprinted here with the permission of the authors and the IEEE.

4.6 Simulation of Communication Networks

By J. F. Kurose and H. T. Mouftah

4.6.1 Introduction

As seen in the previous sections, a tractable analytic model often restricts the range of system characteristics that can be *explicitly* considered in a performance model. For example, systems with complex job scheduling policies, finite buffers (which cause blocking), simultaneous resource possession by a job, complex timing constraints, or tightly coupled interactions between various parts of the model cannot be explicitly modeled using the general-purpose queueing paradigm of the previous sections. In such cases, approximate analytic models may sometimes be adopted [LAVE 83], [SAUE 81] or *simulation* may be used to solve a performance model.

The tradeoffs between an analytic and a simulation approach towards modeling are in the relative amounts of time spent on model formulation and model solution. Since analytic models require a higher degree of abstraction, considerable effort and skill may be required on the part of the network modeler to develop a performance model which accurately reflects the system under study. The analytic model itself, however, can generally be solved rather quickly. In a simulation approach, the system may be modeled to any arbitrary degree of detail; the process model formulation thus becomes a more straightforward task, although it remains advantageous from a solution standpoint to abstract out as many secondary system details as possible. The solution of a simulation model requires significantly more computer time. In some cases, however, simulation is the only viable approach.

The simulation technique typically used to solve a model of a communication network or protocol is *stochastic discrete event simulation.* In discrete event simulation, various components of the actual network under study (e.g., the communication links, buffers, access strategies, network control structures) are represented within a computer program. The *events* that would occur during the actual operation of the network (e.g., the arrival, transmission, routing, and departure of messages, error conditions such as message loss or corruption, link/node failures and recovery) are then mimicked during the execution of the program. The function of the simulation program is thus simply to generate events and then simulate the network's response. (The simulation program typically also performs other ancillary tasks such as recording and later analyzing performance data as well.)

Our goal in this section is not to provide a tutorial introduction to the topic of discrete event simulation; several excellent texts are readily available on both the general topic [FISH 78a], [LAVE 83], [LAW 82], as well as on the use of discrete event simulation for performance modeling, analysis, and design of computer networks [SAUE 83], [SCHO]. Moreover, as discussed in Section VI there are many software tools available which free the practicing modeler from the lower level details of discrete event simulation. Instead, our goal is to provide an overview of several current research areas that are currently advancing the state-of-the-art in using simulation in communication network design and performance analysis.

4.6.2 The Statistical Nature of a Simulation

The generation of events by the simulation program is driven by a stream of *pseudorandom numbers.* These random numbers might be used, for example, to generate the lengths of messages, interarrival times of messages at a given node, time between failures of a link, or probability of a transmission error. Since the event generation process depends on the pseudorandom number stream, the performance measures output by the model (e.g., queueing delays, device utilizations, buffer occupancies, message loss, etc.) are themselves random in nature.

A simulation thus represents a statistical experiment [WELC] and the performance results should thus be subjected to careful statistical analysis. For a given sequence of random numbers, a particular set of values is obtained for the performance measures of interest (e.g., message delay, buffer occupancies, device utilization, throughput, etc.). If the simulation had been run for either more or less time, or if a different stream of random numbers had been used, different values would have been obtained for these performance measures. How then does one determine whether the performance values obtained from a particular run are in some sense the

true or correct values?

1) Transient Versus Steady-State Performance Values: In some cases, the network modeler is interested in performance measures such as the average delay of the first 100 messages through a node (e.g., following a simulated link failure) or the buffer occupancy statistics over some relatively short period of time. In such cases, the performance results may depend quite strongly on the initial conditions of the simulation, e.g., the number of messages initially in the node or buffer when the simulation begins. Performance measures which depend on the initial state of the simulation are referred to as *transient* measures. In general, the amount of simulated time needed to obtain these transient statistics is well-defined (as above) by the nature of transient measures of interest.

In other cases, a modeler may be interested in the long-term or steady-state results of a simulation. In this case, the simulation must typically be run long enough so that the effects of the initial state of the simulation on the performance measures of interest are negligible. Alternatively, the transient portion of a simulation run may be discarded and performance statistics collected only after the simulation has reached steady state. The problem of determining the end of the transient phase is a difficult one, since there is no well-specified point in time at which the transient phase ends. Rather, the effects of the initial configuration become less important as the length of the simulation increases. The problem of identifying the transient phase of a simulation is addressed in [SCHR] and [WELC].

2) Confidence Intervals: Since the performance results predicted by one run of the simulation depend on the particular stream of pseudorandom numbers used to drive the simulation, the results of a performance model will typically vary from one run to another. If we were to run the simulation ten times and obtain ten different values, all within one percent of each other, our confidence in that value would be high. On the other hand, if the ten values obtained varied greatly, our confidence in any one value, or even in the average of the ten values, would be small. In a simulation experiment, *confidence interval techniques* are used to quantify such confidence in a performance estimate. Roughly speaking, if some interval (a,b) is an χ percent confidence interval for the performance measure μ, then if the simulation were to be independently repeated some number of times, the estimated value for μ obtained from the simulation would fall in the interval (a,b) in approximately χ percent of these runs.

Several techniques have been devised for generating confidence intervals. In the method of independent replications, the simulation is run n independent times and n estimates are thus obtained for each performance measure of interest. Each set of n values thus represents n independent samples of the quantity to be estimated and standard statistical techniques can be used to construct a confidence interval [LAW 83], [WELC]. Note

that in the case of steady-state performance measures, the transient portion of *each* of the *n* simulation runs must be discarded, thus making the method of independent replications a potentially expensive confidence interval technique. In the method of batch means, a *single* run of the simulation is divided into N equal-length periods of time (after discarding the initial transient phase). The values of the performance measure during each of these N periods are then taken as N (approximately) independent samples and a confidence interval can again be constructed for this performance measure using standard statistical techniques. Note that this technique has the advantage of discarding just a single transient phase. However, the problem of determining an appropriate value of N is crucial, yet difficult [FISH 78b], [WELC]. If N is too small, the samples will be correlated; if N is too large, an excessive amount of simulation time will be used.

A third technique for generating confidence intervals is known as the regenerative method. This technique is also based on partitioning a single simulation run into independent subruns; it thus also enjoys the advantage of requiring that only a single transient phase be discarded. In the regenerative approach, the simulation run is partitioned on the basis of a modeler-defined *regeneration state,* a state such that the future evolution of the simulation is statistically identical following each entry into this state. For example, in an open queueing model of a network with Poisson arrivals, the regeneration state might be the state with no messages in the system. The regeneration state thus serves to divide the simulation into independent and identically distributed partitions of time of random length. The technique for generating confidence intervals using the regenerative method is described in [CRAN]. One drawback of this method is that a regeneration state may be hard to identify. Also, large models may require an excessive amount of simulation time to pass through enough regeneration points to produce valid confidence intervals.

The three confidence interval techniques discussed above are based on partitioning the overall simulation and obtaining (at least approximately) independent and identically distributed values of performance measures from the partitions. A fourth confidence interval technique, known as the spectral method [HEID 81], is a single run method and explicitly takes into account the correlation between data gathered by the simulation (e.g., between successive queueing times at a queue).

4.6.3 Sensitivity Analysis of Simulation Results

An important part of any modeling study is that of sensitivity analysis--determining how changes in model parameters affect system

performance. For example, a network modeler might be interested in the effect of an increased arrival rate or a decreased channel capacity on the average message delay. Sensitivity analysis also helps identify both critical model parameters, as well as those which have relatively little influence on performance. It thus also provides an indication of the quality and general validity of the model. If performance is extremely sensitive to a certain parameter value, the model may not be applicable over a wide range of parameter value. On the other hand, if performance is insensitive to certain parameters, this would suggest that the model may be needlessly complex and that these parameters might possibly be abstracted out of the model with no loss of applicability.

The most straightforward (and common) approach towards sensitivity analysis is to first run the simulation to determine the baseline performance for the given parameter values. The simulation is then run again several times, each time with a slightly perturbed value of a single model parameter. Clearly, if there are a large number of parameters, this can be a very costly process--if there are N parameters, the simulation must be run N times to determine model sensitivity for the baseline parameter values.

Recently, two approaches have emerged for estimating such sensitivity or gradient information from a *single* run of a simulation. Although their applicability is often restricted to certain classes of models or simulation techniques, they offer a promising methodology for efficiently obtaining sensitivity information. The first approach is known as perturbation analysis (PA) [CASS], [HO 83a], [SURI 87]. One version of this approach, known as infinitesimal perturbation analysis (IPA), is based on the assumption that if an extremely small change had been made in a parameter value before a simulation run, the timing, but not the relative ordering of the simulation events, would have changed. IPA routines can be incorporated into a simulation to track these relative timing changes as the simulation progresses and then produce a sensitivity estimate at the end of the single simulation run. PA has been studied both in the context of single queue systems [SURI 84], [ZAZA] and networks of queues [HO 83b], [HO 84]. Some potential limitations of perturbation analysis are discussed in [HEID 86].

A second approach for obtaining sensitivity estimates from a single simulation run is based on the use of likelihood ratios [REIM]. This technique, which requires only that the occurrence of certain events be counted during the simulation, utilizes the natural variation in the random processes underlying the parameter of interest to generate the sensitivity estimate. In [REIM], it is noted that the likelihood ratio method has an advantage over PA in that sensitivities can be computed with respect to any parameter in a simple and uniform manner. It has the disadvantage, however, of being applicable primarily to regenerative simulation and generally providing sensitivity estimates with a higher variance than those produced using PA.

As discussed above, sensitivity estimates provide important information about the robustness of the simulation results. An additional use of these estimates is in the automated Monte Carlo optimization of the network designs. Typically, network performance results from a complex interaction among many parameters, or design variables, and an explicit closed-form expression seldom exists for the performance measures of interest, much less their derivatives. In such cases, one possible approach is to use simulation to optimize system performance with respect to the design variables. In this case, the system is simulated, sensitivities with respect to the design variables are determined, and the model parameters are then changed in the direction for which the performance increase is greatest. The system is then simulated again and this iterative process repeated until the design has been optimized. Note that the optimization process is complicated by the fact that the gradient values computed via simulation represent noisy estimates of the true gradient values. Hence, a stochastic approximation (optimization) method must be used [GLYN 86b]. A probably convergent simulation-based optimization algorithm is given in [GLYN 86a] which obtains sensitivities using a likelihood ratio approach. The use in optimization of sensitivity estimates via perturbation is studied in [HO 83b].

4.6.4 Speeding Up a Simulation

The major disadvantage of a simulation (as opposed to analytic) approach towards network performance modeling is the amount of time needed to simulate a model. Generally, as network protocols become more sophisticated and complex, so too do their performance models. Moreover, rapidly increasing computation, communication, and switching speeds of communication networks are resulting in ever-increasing message traffic rates. For example, the designs of some fast packet switches, (e.g., [TURN]) permit up to 1.2 million packets to be switched *per second.* Clearly, simulating the operation of even one such switch for any nontrivial amount of time is a formidable computational task.

1) Distributed and Parallel Simulations: One approach towards decreasing the amount of time needed to perform a simulation study is to distribute the computational burden of simulation among several processors in a distributed or parallel multiprocessor system [CHAN 79], [CHAN 81], [COMF], [DAVI], [DECE], [JEFF], [MISR], [PEAC]. In this approach, the simulation is partitioned into numerous *logical processes,* each of which is responsible for simulating one or more of the physical processes in the system being modeled. For example, the operation of each network node

might be simulated by a different logical process; in a high-speed packet switch, a logical process might be responsible for simulating the packet processor for a specific input line, or an element within the switching fabric.

In a multiprocessor environment, different logical processes may be executed in parallel on different processors, with the goal of decreasing the time required until the simulation completes. Of course, processes may not proceed completely asynchronously, since causal interactions between processes in the physical system must also be maintained between the logical processes of the simulation. The manner in which this synchronization is maintained in the simulated system is of fundamental importance, since it is this which prevents a K-fold speedup from being obtained when a simulation is distributed over K processors. Synchronization is typically implemented via message passing between the logical processes. In [DECE], [JEFF], an optimistic approach towards synchronization is taken. Logical processes are permitted to continue a simulation (assuming the absence of a synchronization message) even though a later-arriving synchronization message may require a process to resimulate a previous part of the simulation. This resimulation is accomplished using a simple method for rolling-back the state of the logical process' simulation and undoing its effects on other processes. The experimental results reported in [DECE] suggest that this method yields a linear increase in effective computing power as the number of nodes increases.

In [CHAN 79], [CHAN 81], [MISR], and [PEAC], less asynchrony is permitted but rollback is not required. This approach has been reported to produce significant speedups in the simulation of acyclic queueing networks [MISR]. However, recent studies of actual implementations on a multiprocessor system have shown that for queueing networks in which customer feedback is important (e.g., as in a central server model [SAUE 81]), the synchronization overhead of certain distributed simulation techniques may be so high that simulation distributed over five machines runs several times *slower* than if the simulation had been run on but one of these machines [REED]. A third approach towards distributing a simulation has been investigated in [LUBA]. In this work, a time-driven (as opposed to event-driven) approach was taken, i.e., the simulation clock was incremented by fixed-length time intervals rather than on the basis of event times. For synchronous networks, this was shown to result in a significant speedup of the simulation.

An alternative to distributing a single simulation over K processors is to run K independent copies of the simulation in parallel, one on each of the K processors. In this case, no synchronization is required among the K processors. At the end of the simulation, the results of the K simulations are averaged and the statistical significance of these results determined. As shown in [HEID 85], the simulation termination criteria must be carefully chosen in order to avoid introducing sampling bias into the estimates of the performance measures. A model is also developed in [HEID 85] for comparing distributing a single simulation over K processors with running

K independent copies of simulation on K processors. The purpose of this comparison is to determine which approach is statistically more efficient, i.e., produces performance estimates with a smaller mean squared error for the same amount of computing resources. It is shown that if the run length is long or the initial transient period is short, replicating a simulation is statistically more efficient than distributing a simulation. The question of combining the two methods of simulation is also investigated.

2) Hierarchical Decomposition and Hybrid Techniques: Hierarchical decomposition is a *modeling* technique for speeding up a simulation. In this approach, portions of the model are grouped into submodels and each submodel is replaced by a *single* composite queue with a queue-length dependent service rate. If the submodel has a product form solution (see Section II above), this reduced model is statistically identical to the original model [CHAN 75]. In the case of nearly decomposable systems [COUR 77], the decomposition is approximate, but often quite accurate (see, e.g., the model of SNA virtual route pacing in [SCHW 82]).

The potential speedup results from the fact that the submodel need only be solved once (for each possible customer population) in order to determine the service rates of the composite queue. Then, in the simulation of the entire model, the potentially large number of events that would have occurred when a customer passed through the submodel are replaced by only two events: the arrival and eventual departure of the customer from the composite queue. Experimental results of decomposing extended queueing network models are reported in [BLUM] and several rules of thumb are given for cases in which decomposition would be advantageous.

Note that the submodel itself may be solved either through simulation or analysis, even if the overall model requires a simulation solution. The possibility of such a *hybrid* approach towards modeling [SHAN], i.e., combining both analysis and simulation in a single model, may result in even further reduction in model solution time. Studies examining the use of hybrid models in network modeling can be found in [BHAT], [FROS 86], [SAUE 76], and [VANS]. A hybrid model for studying file transfer protocols in token ring networks was studied in [WONG]. The hybrid model was found to produce performance results typically within 5 percent of those found via a detailed simulation, and did so on the average 50 times faster than the detailed simulation. In the hybrid model of CSMA/CD in [OREI], the operation of a few of the network stations was simulated in detail, while the effects of the remaining stations were modeled by an analytic algorithm which produced sequences of channel busy/idle times. This approach was shown to be more efficient than a detailed simulation of all stations when the overall number of stations in the network was large.

3) Variance Reduction: A third method for decreasing the run time of a simulation is the use of variance reduction techniques. These techniques exploit known statistical properties of the system being modeled in order to

reduce the amount of time needed to obtain performance estimates of a given accuracy. While these techniques have been extensively studied in the literature, they have not often been used in real applications [HEID 84]. An overview of the application of variance reduction techniques in computer network modeling is given in [FROS 88]; a general discussion of variance techniques can be found in [LAW 82]. A promising recent variance reduction technique which has been applied to the simulation of queueing networks is based on the theory of large deviations and is described in [WALR].

To Look Further

The book by Bruell and Balbo [BRUE 80] is an excellent introduction to the convolution algorithm for closed queueing networks. It includes a variety of specialized expressions not covered here as well as performance measures for multiclass networks.

The paper by Reiser [REIS 81] describes the Mean Value Analysis algorithm for state dependent servers. It also relates it to the LBANC algorithm of Chandy and Sauer [CHAN 80] and to the convolution algorithm. The occurrence of overflows in the algorithms is discussed.

Conway and Georganas [CONW 86, 89b] have developed an exact algorithm for multiple chain closed queueing networks which is known as RECAL. It's basis is a new recursive expression which relates the normalization constant of a network with r closed routing chains to those of a set of networks with r-1 routing chains. This algorithm's advantage is that the time and space requirements of the algorithm are polynomial in the number of chains. The algorithm is efficient when there are a large number of chains. By way of contrast, MVA and the convolution algorithm have time and space requirements which increase exponentially with the number of chains. However RECAL's requirements are combinatorial in the number of service centers. The implementation of RECAL on a shared memory multiprocessor is discussed in [GREE].

More recently, Conway, De Souza E Silva and Lavenberg [CONW 89] have developed an algorithm called Mean Value Analysis by Chain (MVAC). The recursion used in this algorithm is different from that used in Mean Value Analysis but similar in structure to the recursion used in RECAL. MVAC does not compute the normalization constant and so it is felt that it avoids the underflow/overflow problems associated with implementing RECAL (or the convolution algorithm for that matter). The computation and storage costs of MVAC are similar to those for RECAL, but different compared to MVA.

Another recent algorithm is Distribution Analysis by Chain (DAC) by De Souza E Silva and Lavenberg [DESO]. It computes joint queue length distributions for product form queueing networks with single server fixed rate, infinite server and queue dependent service centers. It also computes

mean queue lengths and throughputs somewhat more efficiently than RECAL and MVAC.

The usefulness of the PANACEA approach has been extended in a number of papers. State dependent service rates are covered in [McKEN 86]. The calculation of mean queue lengths is discussed in [McKEN 84]. Finally, the use of the PANACEA technique in calculating sojourn time (i.e. time delay experienced by a customer) distribution function is developed in [McKEN 87]. A different piece of work that makes use of asymptotic expansions is [KOGA].

A recent paper of interest on distributed simulation is [WAGN].

Problems

4.1 Consider the convolution algorithm for a single class of customers. Show, as is done in [BRUE 80], that only N+1 memory locations are needed for the operation of the algorithm for both the cases of state dependent and state independent service rates.

4.2 Derive the boxed equation for g(n,m) for state independent servers from the boxed equation before it for state dependent servers.

4.3 Prove that

$$E[n_i] = \sum_{n=1}^{N} np(n_i = n) = \sum_{n=1}^{N} p(n_i \geq n)$$

You can prove this graphically.

4.4 Prove that $g(n,M - \{M\}) = g(n,M-1)$. It should only take two or three lines to do this.

The following problems are designed so that the results can be cross checked with one another.

4.5 Use the convolution algorithm to calculate performance measures for a cyclic network of three queues where $\mu_1 = \mu$, $\mu_2 = 2\mu$ and $\mu_3 = 3\mu$. Specifically:

a) Calculate the normalization constants table and calculate G(N) for N=1,2 ... 5.

b) Calculate the mean throughput, $\overline{\Upsilon}_i(N)$ for N = 1,2 ... 5.

c) Calculate the mean number of customers, $\overline{n}_i(N)$ for i=1,2,3 and N=1,2,3.

d) Calculate the mean time delay through each queue, $\overline{\tau}_i(N)$, for i=1,2,3 and N=1,2,3.

4.6 Use the MVA algorithm to calculate performance measures for the network of the previous problem. Specifically, calculate for i=1,2 ... 5:

a) The mean number in each queue.

b) The mean time delay through each queue.

c) The mean throughput in each queue.

4.7 Use the convolution algorithm to calculate performance measures for a cyclic queueing network of three queues where $\mu_1 = 1.0$, $\mu_2 = 2.0$ and

$\mu_3 = 3.0$. Specifically, calculate:

a) The normalization constants table and the normalization constants G(N) for N=1,2 ... 5.

b) The mean throughput $\overline{\Upsilon}_i(N)$ for N=1,2 ... 5.

c) The marginal queue length distribution when there are four customers in the network using the equation for state independent servers.

d) The marginal queue length distribution for (c) using the equation for state dependent servers. You will need to first calculate $g(N,M-\{i\})$ for N=0,1 ... 4.

e) The utilization, $U_i(4)$ for each queue in the network with four customers.

f) The mean number in the third queue, $\overline{n}_3(4)$ when there are four customers.

g) Check (f) using the marginal probability density.

h) The mean waiting time for the third queue with four customers.

4.8 Use the MVA algorithm to compute performance measures for the network of the previous example. Specifically, calculate:

a) The mean number in each queue.

b) The mean time delay through each queue.

c) The mean throughput in each queue.

4.9 Use the convolution algorithm for state dependent servers to calculate various performance measures for a three queue cyclic network where the service rates for queue 1 are:

Queue 1	
# Customers	Service Rate
1	1
2	5
3	8
4	10
5	11

The service rates for queue 2 are:

Queue 2	
# Customers	Service Rate
1	3
2	5
3	7
4	9
5	11

Queue 3 has a state independent server of rate eleven.

a) Calculate the entries of the normalization constants table. Then calculate G(N) for N=1,2 ... 5

b) Calculate the mean throughput, $\overline{\Upsilon}_i(N)$ for N=1,2 ... 5.

c) Calculate the marginal queue length distribution for the third queue when there are four customers.

d) Calculate, $U_3(4)$, the utilization of the third queue when there are four customers in the network.

e) Calculate, $\overline{n}_3(4)$, the mean number in the third queue when there are four customers in the network using the state dependent formula.

f) Confirm (f) using the marginal state probabilities.

g) Calculate $\overline{\tau}_3(4)$, the mean time delay in the third queue when there are four customers in the network.

4.10 Compare the computational requirements of the solution techniques of equations (4.116) and (4.117) for solving for the equilibrium state probabilities of discrete time queueing systems. Note that solving N general linear equations requires time proportional to the cube of N.

4.11 **Computer Project**: Write and execute a computer program that executes the convolution algorithm for state independent servers. You should be able to input the number of queues in a closed loop and numerical values for the service rates. The program should print out all the performance measures discussed in this chapter. Hand in a report consisting of a program flow chart, the code listing and an example of the program's use.

4.12 **Computer Project**: Write and execute a computer program that executes the convolution algorithm for state dependent servers. You should be able to input the number of queues in a closed loop and numerical values for the service rates. The program should print out all the performance measures discussed in this chapter. Hand in a report consisting of a program flow chart, the code listing and an example of the program's use.

4.13 **Computer Project**: Write and execute a computer program that executes the mean value analysis algorithm for state independent servers. You should be able to input the number of queues in a closed loop and numerical values for the service rates. The program should print out all the performance measures discussed in this chapter. Hand in a report consisting of a program flow chart, the code listing and an example of the program's use.

4.14 **Computer Project**: Write and execute a computer program to solve the system of Figure 4.2 for the equilibrium state probabilities using equation (4.117). Assign initial probabilities to each state uniformly. Include code to renormalize the state probability vector after each iteration. Have the program calculate the throughput of equation (4.118). Produce graphs of throughput for a from .1 to 1.0 in steps of .1 with s=.25,.50,.75. Hand in a report consisting of a program flow chart, the code listing and the graphs.

Chapter 5: Stochastic Petri Nets

5.1 Introduction

Petri Nets provide a means for modeling and graphically representing the possible behavior of systems in which concurrency, serializability, synchronization and resource sharing are important considerations [PETE], [FILM]. They can be used for the understanding and prediction of the behavior of computer systems, communication protocols, biological, economic and other complex systems.

Petri Net models originated from the doctoral dissertation of C.A. Petri [PETR], written in 1962 at the University of Bonn, West Germany and have been extended over the years [BRAU], [DENN], [HOLT]. Such aspects of Petri Nets as determining the set of reachable states have received considerable attention. More recently attention has been given to including appropriate timing mechanisms in Petri Net models.

Stochastic Petri Nets (SPN's) [FLOR] [MOLL 82] are based on the assumption that the time between successive events can be modeled as a random variable. Most work to date has assumed a Markovian statistical framework. We will refer to such Stochastic Petri Nets as Markovian Petri Nets (MPN's). We do not consider the immediate firing transitions of Generalized Stochastic Petri Networks [MARS 84].

The major approaches for the numerical solution of arbitrarily configured Stochastic Petri Nets are the use of a Markov chain solver or simulation. In [DUGA] the features of three existing Stochastic Petri Net solution packages are described and combined into the design of a unified solution package. This package can solve ergodic or transient problems. This includes phase type timing [MARS 85]. The use of regeneration in simulation is discussed in [HAAS].

Work has also been conducted on state aggregation [AMMA], calculating cycle time [WONG CY] and throughput bounds [BRUE 86] [MOLL 86]. Stochastic Petri Net models of data link protocols [MOLL 82], local area networks [GRES] and bus oriented multiprocessor systems [MARS 83] [MARS] have appeared in the literature.

A major difficulty encountered in modeling systems using the Petri Net representation is that the network schematic often becomes unwieldy as the complexity of the system increases. This gives rise to dauntingly large state transition diagrams [DIAZ]. Analytical or even numerical

calculations of equilibrium probabilities become difficult. The actual difficulty, however, is not the large state transition diagram itself, but rather dealing with state transition diagrams with an *arbitrary structure.* For instance, as we have seen in chapter 3, the state transition diagrams of certain classes of queueing networks [BASK 75], [GORD], [JACK 57], [KELL] exhibit enough structure so that simple product form solutions exist for the equilibrium state probabilities. Yet, the state transition diagram can be arbitrarily large or even infinite in extent.

In this chapter a class of Markovian Petri Net models with relevance to practical applications is analyzed. For this class, the state transition diagram of the associated Markov chain exhibits a rich algebraic topological structure that can be readily characterized. Geometrically, the state transition diagram can be embedded in a toroidal manifold. This embedding alleviates the visual clutter of the usual Petri Net schematic diagram. In addition, the equilibrium probabilities satisfy local balance equations and, therefore, exhibit the classical product form. This will serve to reinforce some of the concepts of chapter 3. These results first appeared in, and this chapter follows closely, [LAZA 86, LAZA 87b (copyright 1987 IEEE)].

The chapter is organized as follows. In section 5.2 MPNs are motivated by means of an example of a multiprocessor bus oriented interconnection network. In section 5.3 the general algebraic topological structure of this class of resource sharing protocols is presented and the product form of the equilibrium probabilities derived. In section 5.4 a steady state version of the well known dining philosophers problem is investigated. An aggregate CSMA protocol model appears in section 5.5. The alternating bit protocol is modeled as a Petri Net in section 5.6. Although shorter in length, sections 5.4, 5.5 and 5.6 discuss for their respective settings the theoretical issues raised in the previous sections.

5.2 A Bus Oriented Multiprocessor Model

Consider a dual processor computer system as shown in Figure 5.1. It consists of two processors, P_1 and P_2, and a common shared memory, CM. P_1 or P_2, or both, may attempt transfers to CM through the bus. Conflicts occur as only one processor may utilize the bus, and hence access memory CM, at one time.

We note that this, and the other bus oriented multiprocessor example in this chapter, are based on a modified version of the TOMP architecture [MARS 83]. TOMP is the TOrino MultiProcessor developed at the Politecnico of Torino in Torino, Italy.

A Markovian Petri Net model of this system appears in Figure 5.2. Formally, the MPN is defined as the sixtuple:

$$\mathbf{P} = (P, T, I, O, M, Q) \tag{5.1}$$

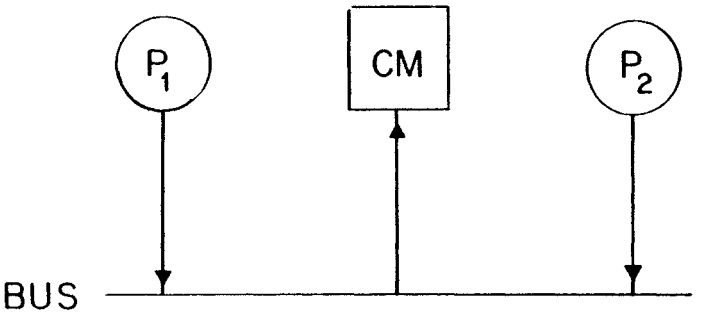

Fig. 5.1: Dual Processor, Single Memory, Multiprocessor

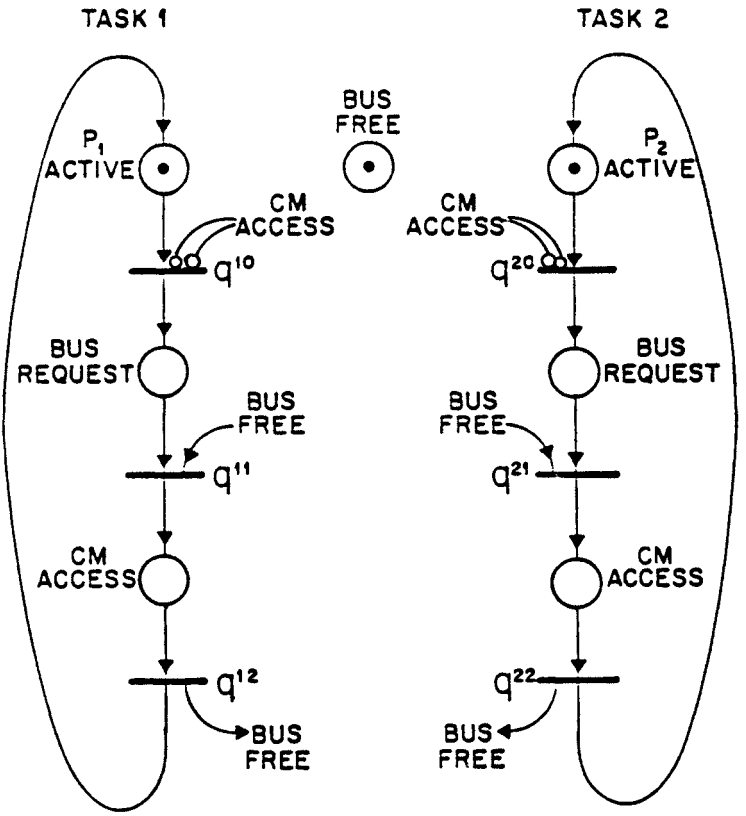

Fig. 5.2: Petri Net of Fig. 5.1

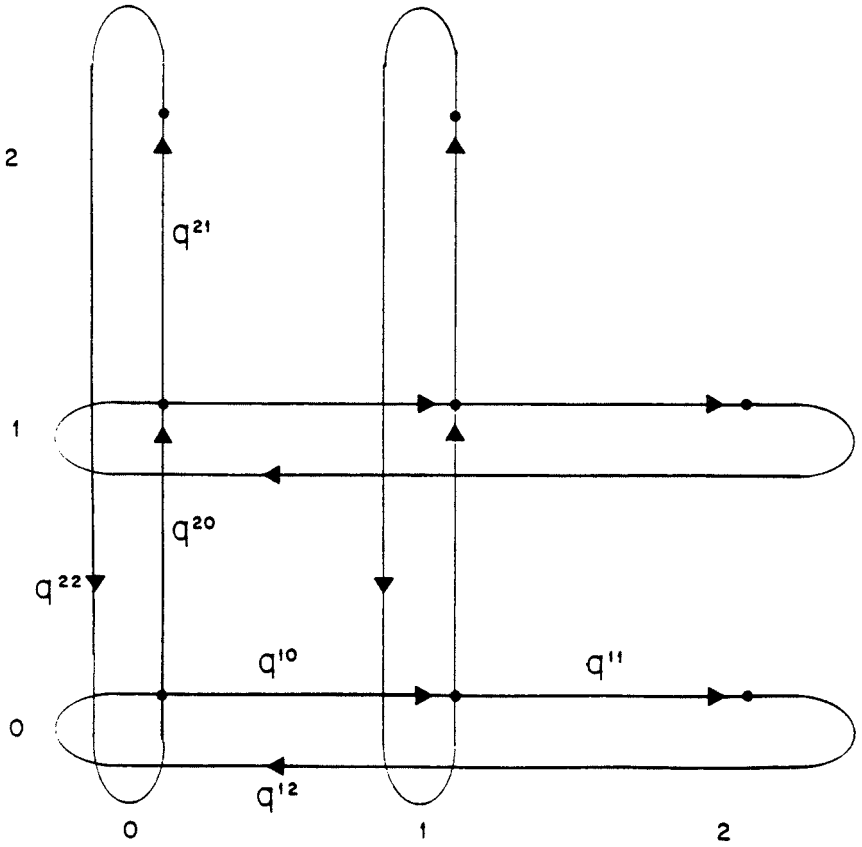

Fig. 5.3: State Transition Diagram of Fig. 5.2

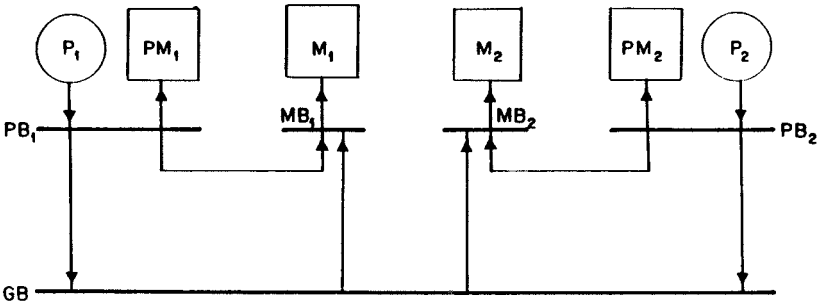

Fig. 5.4: Dual Processor, Dual Memory, Multiprocessor

Copyright 1987 IEEE

P is a set of places (drawn as circles) and T is a set of transitions (drawn as horizontal bars). I is an input function which maps each transition to one or more places (drawn as directed arcs) and O is an output function which maps each place to one or more transitions (drawn as continuous or dotted directed arcs). A marking, M, is an assignment of a non-negative number of tokens (drawn as dots) to individual places. For a given Petri Net, the marking defines the state of the system. Q is the set of rates associated with the transitions.

The rules of Petri Net operation are simple. If there is at least one token in each place incident (through solid or dotted arcs) to a transition, that transition is enabled for firing. This involves removing one token from each place incident to the transition (except from places connected to the transition with dotted arcs) and adding one token to each place connected to the transition by an outgoing arc. The use of these non-standard dotted arcs will become clear as we proceed. For a MPN the time between an enabled transition and its firing is an exponentially distributed random variable. In a deterministically timed Petri Net this time would be a deterministic quantity (zero for an "immediate" transition [MARS 84]).

Note that the presence of a small circle where an arc meets a transition indicates a complemented dependency. For instance, in Figure 5.2 one can move to a BUS REQUEST place only if P1 is ACTIVE and there is *no* ongoing CM ACCESS.

The class of MPN's considered in this chapter is safe, that is, a place may hold at most one token at any given time. As we are interested in steady state solutions all transitions also exhibit liveness, *i.e.*, deadlock cannot occur.

In the dual processor MPN model of Figure 5.2 one can see a total of two linear sequences of events originating from the places indicating that the processors are internally active (externally idle). Each such sequence will be referred to as a *task sequence* and the events which comprise it as *sub-tasks.* The first state is often an idle event (*i.e.*, task waiting to be initiated). The two tasks of Figure 5.2 thus correspond to P_1 or P_2 attempting to perform a transfer with CM.

Generally, one can distinguish between places which represent subtasks and those which represent resources (*e.g.*, GB FREE). *The essence of the Petri Net paradigm is that a task sequence may be blocked (temporarily or permanently) if resources are not available.* The resource places and associated transitions could be replaced by a set of control rules making enablement dependent on the state of selected task sequences. In [GHAN] a similar distinction is made between "job tokens" and "control tokens".

In Figure 5.2, q^{kl} is the exponential rate associated with the l th subtask ($l = 0,1,2$) of the k th task sequence ($k=1,2$). The Petri Net transitions should not be confused with the transitions of the state transition diagram of the Markov process associated with the Petri Net. The corresponding state transition diagram appears in Figure 5.3. The

horizontal axis corresponds to P_1 and task sequence 1 while the vertical axis corresponds to P_2 and task sequence 2. The equilibrium state probabilities are given below for all states (i,j) of the indicated task sequences:

$$p(i,j) = \frac{q^{10}\ q^{20}}{q^{1i}\ q^{2j}}\, p(0,0) \tag{5.2}$$

Note that these probabilities have the characteristic product form which is known to exist for certain classes of queueing networks.

Consider now the more sophisticated multiprocessor system shown in Figure 5.4. It consists of two processors, P_1 and P_2 and the corresponding memories, M_1 and M_2. P_1 (or alternately, P_2) may attempt transfers to $M_1(M_2)$ through the local memory bus $MB_1(MB_2)$ connected to the private processor bus $PB_1(PB_2)$. Alternatively, $P_1(P_2)$ may attempt transfers to $M_2(M_1)$ via the global bus (GB) and the local memory bus $MB_2(MB_1)$. Conflicts occur as only one processor may utilize the local bus or the global bus at a time. Note that the processor memories, PM_1 and PM_2, are only accessible from their respective processors on the private busses PB_1 and PB_2, respectively. Unlike in [MARS 83], we assume that processor $P_1(P_2)$ attempts an access only when the local bus, and if necessary the global bus, are free. As will be seen, this necessary to guarantee the existence of the product form solution.

In the multiprocessor MPN model of Figure 5.5 one can see a total of four task sequences of events originating from the places indicating that the processors are (internally) active or idle with respect to the rest of the system. The four task sequences of Figure 5.5 correspond to P_1 or P_2 attempting to perform a transfer with M_1 or M_2.

In Figure 5.5, q^{kl} is the exponential rate associated with the l th subtask (l = 0,1,2,...) of the k th task sequence (k=1,2,3,4). The corresponding state transition diagram appears in Figure 5.6. The horizontal axis corresponds to P_1 and task sequence 1 and 3 while the vertical axis corresponds to P_2 and task sequence 2 and 4. The equilibrium state probabilities, for each quadrant of the diagram, are given below for all states of non-zero probability. For all states (i, j) of the indicated task sequences we have:

$$p(i,j) = \frac{q^{10}\ q^{20}}{q^{1i}\ q^{2j}}\, p(0,0) \tag{5.3}$$

for tasks 1 and 2,

$$p(i,j) = \frac{q^{30}\ q^{40}}{q^{3i}\ q^{4j}}\, p(0,0)$$

for tasks 3 and 4,

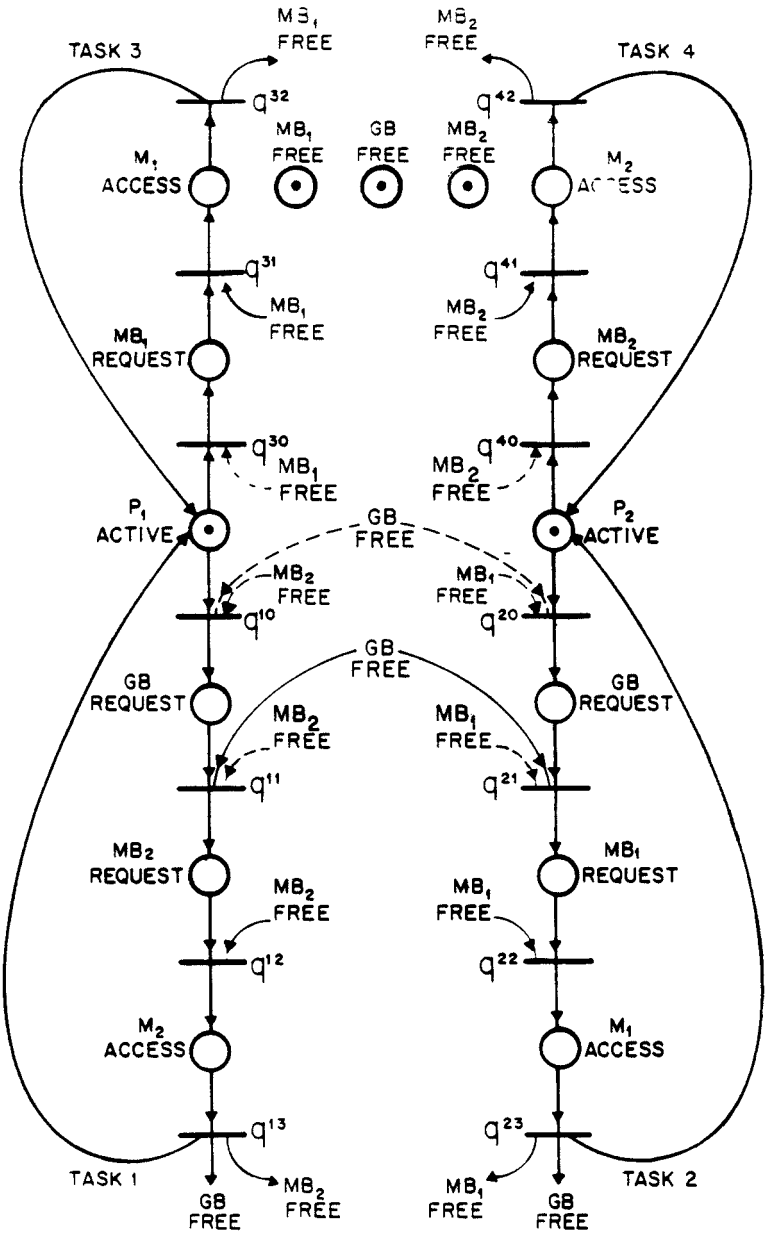

Fig. 5.5: Petri Net of Fig. 5.4

Copyright 1987 IEEE

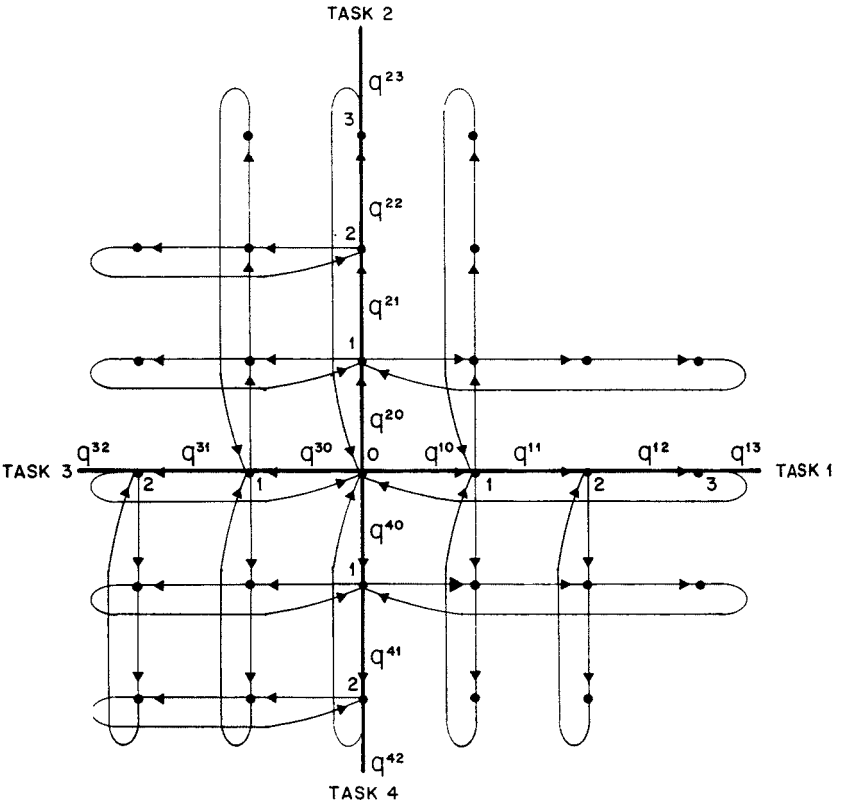

Fig. 5.6: State Transition Diagram of Fig. 5.5

Copyright 1987 IEEE

$$p(i,j) = \frac{q^{10}}{q^{1i}} \frac{q^{40}}{q^{4j}} p(0,0)$$

for tasks 1 and 4, and

$$p(i,j) = \frac{q^{20}}{q^{2i}} \frac{q^{30}}{q^{3j}} p(0,0)$$

for tasks 2 and 3.

Note that these probabilities have the characteristic product form which is known to exist for certain classes of queueing networks. The basis of the surprising result of (5.2) and (5.3) will be explained in the next section in terms of the algebraic topological structure of the state transition diagram of the Markov chain.

5.3 Toroidal MPN Lattices

To understand why the MPN of Figure 5.2 and Figure 5.5 has a product form solution, consider two *independent* task sequences. That is, sufficient resources exist so that the state of each sequence does not affect the state of the other sequence. This leads to a state transition diagram like that of Figure 5.7. Let task 1 correspond to the horizontal axis and task 2 correspond to the vertical axis.

While the position of the states appears to conform to a rectangular geometry, when one considers the "wrap around" character of the boundary transitions it becomes apparent that the natural topological space [MASS] for embedding the state transition diagram is the surface of a torus. This toroidal embedding is natural in that it is symmetrical and planar. The class of MPNs that can be naturally embedded on a torus will be referred to as *Toroidal Markovian Petri Networks.*

The torus, a two dimensional manifold [MASS], can be thought of as a smooth surface without a boundary. The state transition diagram of the dual processor system of Figure 5.1 and Figure 5.2 embedded in a torus in R^3 is shown in Figure 5.8. The embedding of the four quadrant state transition diagram of Figure 5.6 into torii is shown in Figure 5.9. Only the cycles used for pasting torii together are shown here. In this pasting the top and bottom torii rest upon each other and the left and right torii pass partially through each other.

It is apparent in Figure 5.7 that the global balance equations at each point in the state transition diagram are structurally the same. Boundary states, which often appear in queueing based state transition diagrams, need not be considered separately.

While any state may be chosen as a reference state, in applied terms the natural choice is the state which corresponds to both independent tasks being active (idle). With this state being given by the coordinates (0,0), we

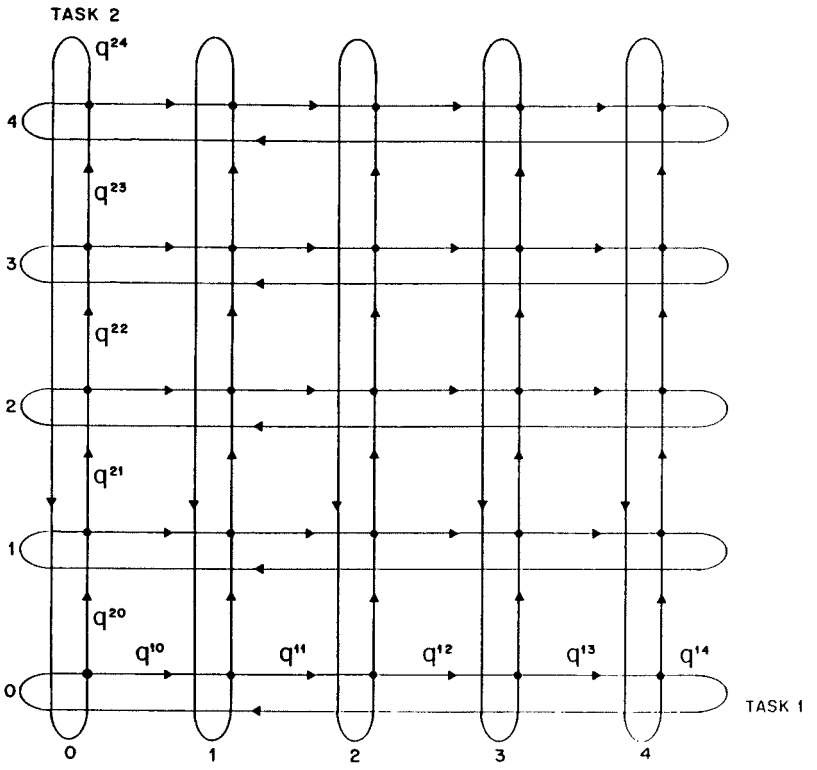

Fig. 5.7: State Transition Diagram, Two Independent Processors

Copyright 1987 IEEE

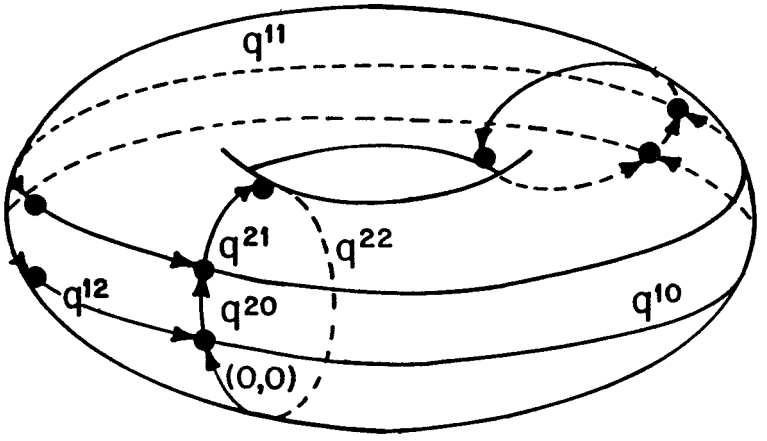

Fig. 5.8: Transition Diagram of Fig. 5.3 Embedded in Torus

seek an expression for equilibrium probabilities $p(i,j)$. The global balance equation for the conservation of probability flux at vertex (i,j) is:

$$(q^{1i} + q^{2j})\, p(i,j) = q^{1,i-1}p(i-1,j) + q^{2,j-1}p(i,j-1) \tag{5.4}$$

As has been mentioned in chapter 2, previous work [LAZA 84a,b,c], [WANG 86] on the algebraic topological structure of queueing network state transition diagrams has shown that the classical Jackson-type product form solution of the equilibrium probabilities can be traced back to a decomposition of the state transition diagram into elementary building blocks. For a class of state transition diagrams [WANG 86], cyclic flows can be associated with these building blocks. A cyclic flow is associated with the transitions of each building block and has equal magnitude of probability flux on each transition. Note that, the state transition diagram structure in Figure 5.7 suggests the presence of cyclic flows about the torus (*e.g.*, (0,0), (1,0), (2,0), (3,0), (4,0) to (0,0)). It will be shown that these cycles are the fundamental building blocks [LAZA 84a] of the state transition diagram.

This observation is equivalent (Duality Principle [WANG 86]) to the hypothesis that the following local balance equations are satisfied :

$$q^{1,i-1}p(i-1,j) = q^{1i}p(i,j) \tag{5.5}$$

$$q^{2,j-1}p(i,j-1) = q^{2j}p(i,j)$$

These local balance equations are suggestive of a natural recursion for $p(i,j)$:

$$p(i,j) = \frac{q^{10}\ q^{20}}{q^{1i}\ q^{2j}}\, p(0,0) \tag{5.6}$$

This does indeed satisfy the global balance equations (5.4). For N independent tasks the expression can be generalized in a straight forward manner to :

$$P(k_1,k_2,\cdots,k_N) = \prod_{l=1}^{N} \frac{q^{l0}}{q^{lk_l}}\, p(0,0,\ldots,0) \tag{5.7}$$

where k_l is the k_l th sub-task of the l th task sequence. That we have achieved a product form solution is not surprising at this point since task independence was assumed.

This is indeed the case in the lower left quadrant of Figure 5.6 which corresponds to concurrent access by each processor to its corresponding memory. What of the other quadrants? Each can be seen to have the same structure as the state transition diagram of Figure 5.7 *with specific cyclic building blocks removed.* The removal of a building block(s) does not affect the form of the solution, except to cause a renormalization of p(0,0,...,0). Consistency of the local balance equations is preserved since a form of geometric replication applies [LAZA 84a,c]. Therefore, as has been discussed in chapter 2, the flow on a cycle is isolated [WANG 86], *i.e.*, it neither contributes nor removes probability flux from the other parts of the state transition diagram.

The remaining question to be answered is why the *pasting* (an algebraic topological concept) of the four quadrant state transition diagrams of Figure 5.6 preserves consistency and the product form solution. Note that the boundary between any two quadrant state transition diagrams consists of a cycle (see Figure 5.9). For each of the quadrants taken in isolation, the equilibrium probabilities of the states of this cycle have the same product form when expressed as a function of an arbitrary state of the cycle. This property preserves, therefore, the *continuity* of the form of the state probabilities. Thus, when two quadrant state transition diagrams are pasted together (see Figure 5.9), the form of the solution of the equilibrium probabilities for each quadrant remains the same, except for the renormalization factor.

We can summarize the properties of the MPN investigated above as:

Theorem: A safe MPN consisting of a number of task sequences that are comprised of a series of sequential sub-tasks has a product form solution for the equilibrium state probabilities if the state transition lattice can be naturally associated with a Cartesian coordinate system (alternatively: naturally embedded into an aggregation of toroidal manifolds) and if the state transition lattice is comprised of integral building blocks and corresponding consistent set of local balance equations.

Proof: As explained above, the assumed solution form is verified to be a solution of the system's local balance equations.

The properties of the MPN investigated above have been characterized in terms of the algebraic topological structure of the state transition diagram of the associated Markov chain. This characterization can be translated into the Petri Net schematic representation as follows:

Theorem: Consider a safe Markovian Petri Net consisting of a number of task sequences that are comprised of a series of sequential sub-tasks. The product form solution for the equilibrium state probabilities exists, if and only if, a task sequence is only allowed to proceed if there is a non-zero

probability that it can return to its current state without the need for a state change in other task sequences.

Proof: This characterization is equivalent to the state transition lattice being comprised of integral building blocks.

Essentially, the characterization precludes the possibility of a form of blocking. It is well known that the product form solution does not exist in blocking environments. This is a fundamental limitation that the algebraic topology of the state transition lattice imposes. In short, this characterization is useful for determining which models do and do not have a product form solution.

As an example, note that the MPN of Figure 5.5 differs from the more practically oriented one in [MARS 83] in the presence of the dotted arcs in the Petri Net schematic. These provide additional restrictions on when the task sequence is allowed to proceed. Without this restrictions the state transition diagram of Figure 5.6 gains six transitions:

$$(3,0)\rightarrow(3,1) \qquad (0,3)\rightarrow(1,3)$$
$$(2,0)\rightarrow(2,1) \qquad (0,2)\rightarrow(1,2)$$
$$(3,0)\rightarrow(3,-1) \qquad (0,3)\rightarrow(-1,3)$$

which do not belong to a cycle and do not fit into the above characterization. These transitions modify the circulatory pattern of the state transition diagram and the classical Jackson-type product form solutions does not hold for this case.

The discussion so far, pointed out a general methodology for determining whether the equilibrium probabilities have a product form that can be obtained by inspection. Consequently, the set of equations (5.7) holds for a multiprocessor system with an arbitrary number of processors, as long as its state transition diagram can be constructed using the previous simple rule.

In short, this characterization is useful for determining which models do and do not have a product form solution. It is somewhat disappointing that the product form solution does not exist in the presence of blocking, but this is also the same situation that occurs in queueing networks. This is a fundamental limitation that the algebraic topology of the state transition diagrams imposes upon us.

5.4 The Dining Philosophers Problem

This well known model, due to Dijkstra [DIJK], is a good example of a distributed protocol for shared resources and has often appeared in Petri Net form [PETE], [ZENI]. A number of philosopher's, say five, are seated

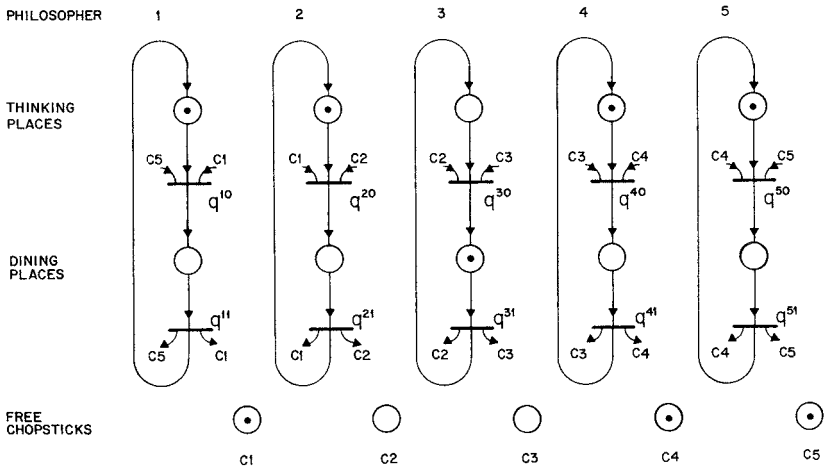

Fig. 5.10: Dining Philosophers Petri Net

Copyright 1987 IEEE

around a circular table and are dining with chopsticks. A *single* chopstick is placed on the table between each philosopher. Naturally, if a philosopher picks up the chopsticks on either side of him (her) to dine with, the philosopher's neighbors may not dine.

A MPN model of the dining philosophers problem, with five philosophers, is depicted in Figure 5.10. This is a steady state version and is thus deadlock free. The task sequence for each philosopher can be seen to consist of simply an idle and a dining state.

In the state transition diagram of the associated Markov chain (see Figure 5.11) there are five states which correspond to a single philosopher dining and five states which correspond to the maximum of two philosophers dining. The reference state of the state transition diagram corresponds to all philosophers being in the thinking state. This is a sufficient state description since the system is memoryless.

In states which correspond to only one philosopher dining, the only task sequences which may be initiated are those which correspond to the other two *allowable* philosophers making a transition into the dining state.

Let q^{k0} be the rate at which the k th philosopher makes a transition to the dining state and q^{k1} be the rate at which the same philosopher leaves it. Then, over the set of allowable states:

$$p(k,k\pm2) = \frac{q^{k\pm2,0}}{q^{k\pm2,1}}\,p(k) = \frac{q^{k\pm2,0}}{q^{k\pm2,1}}\,\frac{q^{k0}}{q^{k1}}\,p(0) \tag{5.8}$$

for all k, $1 \leq k \leq 5$, where $\pm$ is modulo 5 addition and subtraction.

This result is the direct consequence of the algebraic topological structure of the state transition diagram. The "wrap around" character of the state transition diagram depicted in Figure 5.11 leads to an embedding of it into a five dimensional toroidal manifold.

(An N-dimensional manifold exhibits locally the properties of the N-dimensional Euclidean space. A standard technique to visualize a multidimensional torus is by topological identification. For example, a 2-dimensional torus can be obtained by identifying opposing sides of a square as being equivalent. A 3-dimensional torus is equivalent with a cube with the opposite planes identified as being equivalent [MASS], [FIRB].)

Generalization of the dining philosophers problem to an arbitrary number of philosophers is straightforward. The natural space to embed the state transition diagram is the N-dimensional toroidal manifold, where N is the number of philosophers. Since geometric replication applies, a simple product form solution of the type given by equation (5.8) holds.

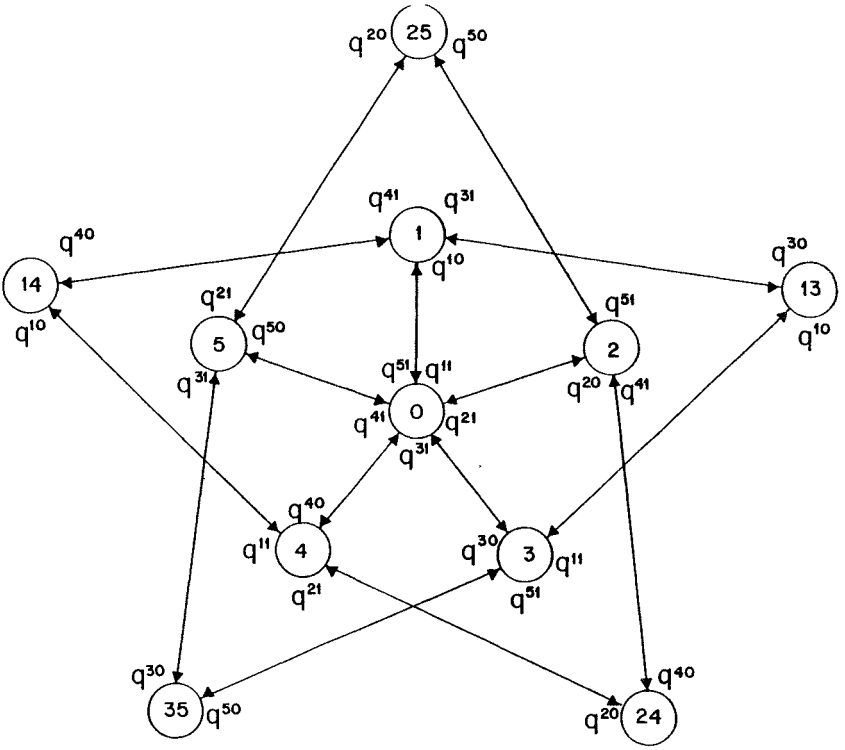

Fig. 5.11: State Transition Diagram of Fig. 5.10

Copyright 1987 IEEE

5.5 A Station Oriented CSMA Protocol Model

Since bus oriented multiprocessor architectures can be modeled by Petri Nets, the modeling of bus type local area networks is another potential application. Petri Net models of bus type local area networks have appeared in [GRES], [MOLL 84].

Bus type local area network models may either attempt to model the various stations and their interaction in detail or to model the behavior of a single station in the context of the aggregate statistical behavior of the remainder of the network. A CSMA model of the latter type will now be introduced. It should be noted that the difficulty in taking the former approach for local area networks, as opposed to multiprocessor bus interconnections, lies in the non-negligible propagation delay along the bus.

A MPN of the LAN station appears in Figure 5.12 and the corresponding state transition diagram in Figure 5.13. Messages originate at the station with rate q^{10}. They are transmitted after an average time of $1/q^{11}$ spent in channel scanning. Once transmission has been initiated, collisions occur with probability ϵ [SHAP] after an average time of $1/q^{12}$. The average duration of a successful transmission is $1/q^{12} + 1/q^{13}$. The average collision resolution time is $1/\tau$.

Note that the q^{11} transition in the Petri Net does not depend on the bus availability. While, bus availability could be modeled stochastically, more correctly it depends on the states of other stations and their connection through the bus with significant propagation delay. The simplified model of Figure 5.13 will be used to illustrate some relevant ideas concerning the toroidal class rather than as a detailed representation of the system operating under the CSMA protocol.

The equilibrium probabilities of the states along the state transition diagram's main loop are:

$$p(k) = \frac{q^{10}}{\alpha q^{1k}} p(0) \tag{5.9}$$

where α is $(1-\epsilon)$ when k represents the MESSAGE READY and START TRANSMIT states and 1 otherwise. One way of viewing the state transition diagram of Figure 5.13 is as a cycle embedded in a one dimensional toroidal manifold with a "feedback" loop attached which corresponds to the possibility of collisions. The product form is maintained since the split of probability flux between the two loops is proportional to $1-\epsilon$ and ϵ.

However, when searching for a natural topological space for embedding the entire state transition diagram must be considered. While this state transition diagram can be embedded on a 2-d torus, this could also be accomplished using a 2-d spherical manifold. A circle is a one dimensional toroidal manifold as well as a degenerate one dimensional spherical

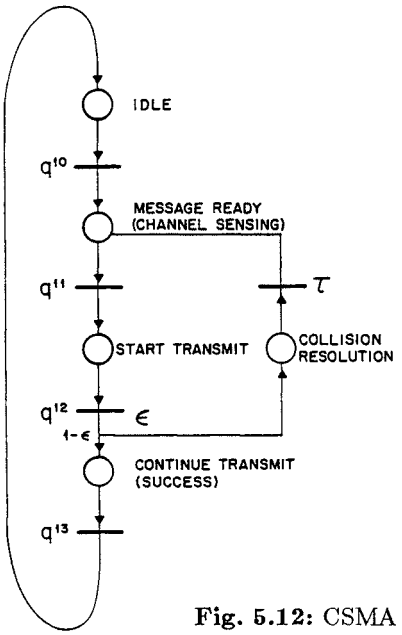

Fig. 5.12: CSMA Petri Net

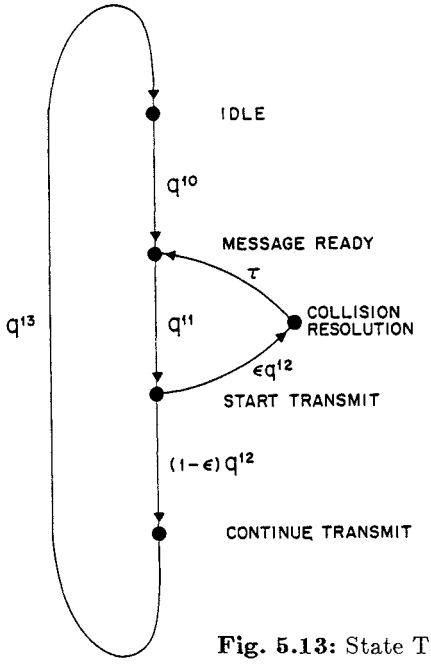

Fig. 5.13: State Transition Diagram of Fig. 5.12

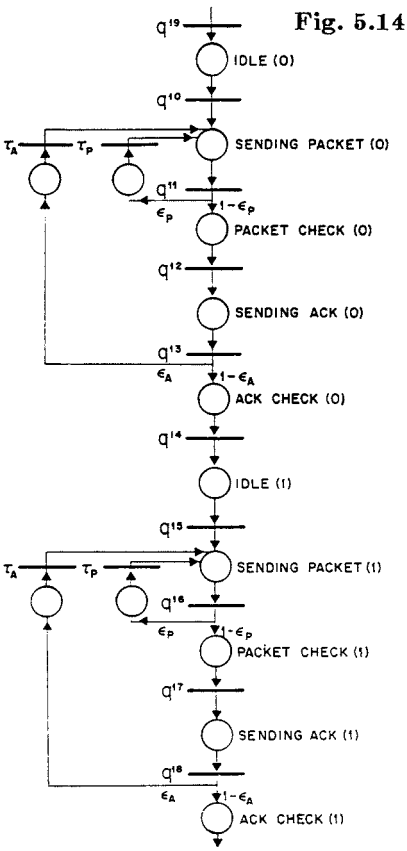

Fig. 5.14: Alternating Bit Protocol Petri Net
Copyright 1987 IEEE

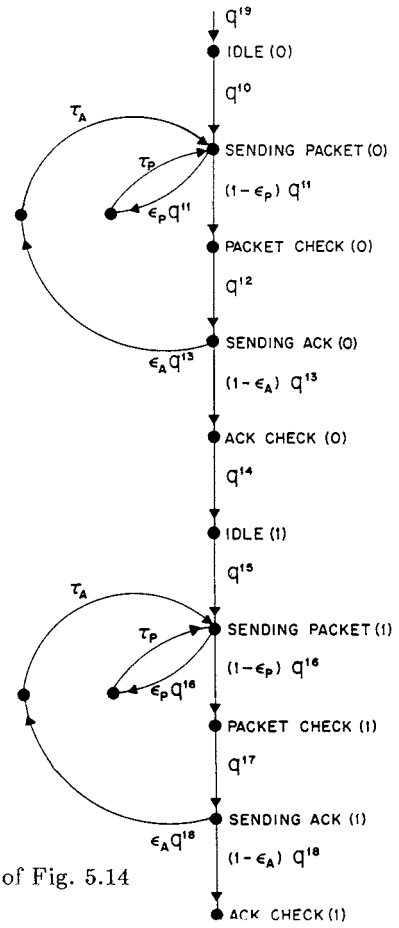

Fig. 5.15: State Transition Diagram of Fig. 5.14
Copyright 1987 IEEE

manifold.

The key point in the analysis above is the random bifurcation of flows. Thus it would seem most appropriate to view the state transition diagram as being embedded on a manifold obtained by the "pasting together" (an algebraic topological concept) of two 1-d toroidal manifolds. The pasting occurs along shared transitions and has been previously noted for queueing networks with random routing [LAZA 86]. A detailed example of pasting along nodes appears in [LAZA 84a,b,c], [WANG 86].

5.6 The Alternating Bit Protocol

The alternating bit protocol has been previously modeled in state machine [KRIT] and Petri Net form [MOLL 82]. A bit is alternatively set to 0 or 1 in successive packets in an acknowledgement and time-out based link. The alternating bit allows the receiver to disregard packets which are erroneously retransmitted due to acknowledgement loss.

A MPN of the protocol appears in Figure 5.14 and the corresponding state transition diagram in Figure 5.15. Note that the state transition diagram consists largely of a single cycle with some minor feedback loops. Token movements off the main cycle occur with probability $1-\epsilon_P$ and $1-\epsilon_A$ [SHAP]. Here the probability of packet loss is ϵ_P and that of acknowledgement packet loss is ϵ_A. The nominal task sequence consists of starting from the idle state (with the acknowledgement bit at 0 or 1) and then sending the packet, checking the correctness of the received packet, sending an acknowledgement packet and checking the correctness of the acknowledgement packet. These tasks are then repeated with the bit complemented.

The loops are due to either the loss in transit through the packet network of either the packet or the acknowledgement packet. In the alternating bit protocol a timer at the transmitter, started after packet transmission, indicates when an acknowledgement is overdue ("time out"). After time out a retransmission occurs. The states off the main cycle help to model this timeout mechanism.

Let τ_P and τ_A be the rate modeling the average timeout duration for message and acknowledgement packet loss, respectively (see Figure 5.14). In Figure 5.15, τ_P originates from a state corresponding to packet loss; there is also a state with outgoing rate τ_A modeling timeouts due to acknowledgement packet loss. The transmitter timer can not distinguish between message packet and acknowledgement packet loss so that the average timeout setting must be the same in either case. Thus $1/\tau_P = 1/\tau_A + 1/q^{12} + 1/q^{13}$.

The equilibrium probabilities along the main cycle have the form :

$$p(k) = \frac{q^{10}}{\alpha\ q^{1k}} p(0) \qquad (5.10)$$

where α is equal to $(1-\epsilon_P)\times(1-\epsilon_A)$ or $(1-\epsilon_A)$, depending whether the state k represents the SENDING PACKET, or the PACKET CHECK and SENDING ACK states, respectively. For all other states belonging to the main cycle $\alpha=1$.

The equilibrium probabilities of the states in the feedback loops in Figure 5.15 can be also easily found. The product form of a single cycle is maintained, in spite of the feedback loops, because the splits of flow are proportional to ϵ_P and $(1-\epsilon_P)$ and to ϵ_A and to $(1-\epsilon_A)$. Again, a number of one dimensional toroidal manifolds are pasted together to embed the state transition diagram. Although the two outer feedback loops in Figure 5.15 change the character of the embedding manifold, the product form of the equilibrium probabilities still applies.

5.7 Conclusion

A class of Markovian Petri Net models with equilibrium probabilities that can be expressed in the classical product form has been described. Examples have been given involving resource sharing (a bus oriented system and the dining philosophers problem) as well as communication protocol problems (the alternating bit protocol) that arise in practical applications.

The results obtained here have been based on the algebraic, geometric and topological properties of the associated state transition diagrams. The basic principle derived represents a structural result characterizing the existence of Markovian models for resource sharing having simple product form equilibrium probabilities.

The wide spread use of Petri Net models has been somewhat inhibited by a perceived unwieldiness of the Petri Net schematic. It has been found by the authors of [LAZA 86,87b] that drawing the task sequences in a linear form and not completely drawing directed arcs to and from resource places tends to enhance the clarity and readability of Petri Net schematics. While such a structured approach has sometimes been implicit in previous work, this work provides an explicit basis for its use.

To Look Further

The characterization of the product form solution for stochastic Petri Nets discussed in this chapter can be extended to the case where the enabled transitions fire after a time that is a sum of exponential random variables [WANG 89].

An excellent source for material concerning timed Petri nets are the proceedings of a series of international workshops that have been held on

the subject. The first was the International Workshop on Timed Petri Nets, held in Torino, Italy during 1985 (IEEE Computer Society Press). The second is the '87 International Workshop on Petri Nets and Performance Analysis held in Madison, Wisconsin USA in 1987 (IEEE Computer Society Press). The last is the 3rd International Workshop on Petri Nets and Performance Models held in Kyoto-shi, Japan during December 1989 (IEEE Computer Society Press).

A recently developed package for numerically solving stochastic Petri Net problems is SPNP (Stochastic Petri Net Package). It was developed at the Computer Science Dept. of Duke University, Durham N.C., by G. Ciardo, J. Muppala and K.S. Trivedi. Based on C, it runs on a variety of systems. It is capable of solving generalized stochastic Petri nets in both the transient and steady state case. For details see [CIAR]. This article includes a comparison with two other packages: GreatSPN [CHIO] from the University of Torino, Italy and METASAN [SAND] from the University of Michigan at Ann Arbor and Industrial Technology Institute, Ann Arbor. These latter two packages have simulation capabilities.

Appendix: Probability Theory Review

A.1 Probability

Probability theory is ultimately concerned with the results of statistical "experiments". An experiment will be assumed to have a number of outcomes which can not be further decomposed [FELL]. For instance, if we pick a card from a randomly shuffled deck of cards there are 52 possible outcomes. Each such outcome is a **sample point**. The collection of such sample points is the **sample space**, S. It is assumed that the sample points are mutually exclusive.

Sample points can be aggregated into subsets of the sample space called **events**. For instance, in picking a card in the previous example, two potential events are the selection of a red card and the selection of a black card.

A **probability measure** defined on S is an association of real numbers to the events, which satisfies the following axioms [THOM] [KLEI 75]:

$$P(S) = 1.0 \tag{A.1}$$

$$0 \leq P(A) \leq 1 \quad \text{for } \textit{any event } A \tag{A.2}$$

and if A and B are mutually exclusive:

$$P(A + B) = P(A) + P(B) \tag{A.3}$$

These equations are the axiomatic way to define probability [PAPO]. One can also define it, intuitively, in terms of relative frequency. If an event A occurs n_A times out of n tries then:

$$P(A) = \lim_{n \to \infty} \frac{n_A}{n} \tag{A.4}$$

Let P(A|B) be the **conditional probability** of event A occurring in an experiment given that B *has* occurred. Let P(AB) be the probability of the intersection of events A and B. Then:

$$P(AB) = P(A)P(B \mid A) = P(B)P(A \mid B) \tag{A.5}$$

Two events are **statistically independent** if:

$$P(B \mid A) = P(B) \text{ and } P(A \mid B) = P(A) \tag{A.6}$$

This, together with (A.5), implies that for two statistically independent events:

$$P(AB) = P(A)P(B) \tag{A.7}$$

For instance, let A be the event of drawing a diamond and B be the event of drawing a red card. Clearly, P(A)=.25, P(B)=.5, P(A|B)=.5, P(B|A)=1.0, P(AB)=.25. However, P(AB)≠P(A)P(B) as these events are *not* independent.

In general, for independent events A,B,C...

$$P(ABC\ldots) = P(A)P(B)P(C)\ldots \tag{A.8}$$

P(ABC...) is known as the **joint probability** and P(A) is an example of a **marginal probability**.

A.2 Densities and Distribution Functions

Suppose that we are interested in a continuous random variable, X, that assumes real values. Then the **cumulative distribution function** [THOM] is $F(a) = P(X \leq a)$ for argument a. Among the properties of the cumulative distribution function are:

$$F(a) \geq 0 \tag{A.9}$$

$$F(-\infty) = 0, \quad F(\infty) = 1 \tag{A.10}$$

If $a_1 < a_2$ then:

$$F(a_2) - F(a_1) = P(a_1 < x < a_2) \tag{A.11}$$

Finally, the cumulative distribution function is non-decreasing as a function of a:

$$F(a+\Delta) \geq F(a) \quad \text{if } \Delta > 0 \tag{A.12}$$

A **probability density function** is related to the cumulative distribution function by

$$f(x) = \frac{dF(X)}{dx} \tag{A.13}$$

or:

$$F(x) = \int_{-\infty}^{x} f(y)dy \tag{A.14}$$

For a continuous probability density function:

$$f(x) \geq 0 \tag{A.15}$$

$$\int_{-\infty}^{\infty} f(x)dx = 1 \tag{A.16}$$

$$P[a_1 < x \leq a_2] = \int_{a_1}^{a_2} f(x)dx \tag{A.17}$$

Now consider a probability density function on a discrete state space. Then:

$$f(x_i) \geq 0 \tag{A.18}$$

$$\sum_{x_i \epsilon S} f(x_i) = 1 \tag{A.19}$$

$$P[a_1 \leq x \leq a_2] = \sum_{i=a_1}^{a_2} f(x_i) \tag{A.20}$$

Example: The negative exponential density is used throughout this text:

$$f(x) = \mu e^{-\mu x} \qquad x \geq 0$$

The cumulative distribution function is:

$$F(x) = \int_0^x \mu e^{-\mu y}\, dy = 1 - e^{-\mu x}, \quad x \geq 0$$

A.3 Joint Densities and Distributions

The **joint distribution function** of random variables X_1 and X_2 is

$$F(x_1,x_2) = P[X_1 \leq x_1,\ X_2 < x_2] \tag{A.21}$$

for arguments x_1 and x_2 and the **joint density function is:**

$$f(x_1,x_2) = \frac{d^2F(x_1,x_2)}{dx_1 dx_2} \tag{A.22}$$

These two expressions can be extended, in the natural way, to situations involving more than two random variables.

The **marginal density function** is related to the joint density function through:

$$f(x_1) = \int_{-\infty}^{\infty} f(x_1,x_2)dx_2 \tag{A.23}$$

That is, if the joint density function is integrated out over x_2, what is left is $f(x_1)$. For a joint distribution of n variables, one integrates out over all but the ith variable to obtain $f(x_i)$.

The **conditional probability density** is related to the joint and marginal probability densities through:

$$f(x_1 \mid x_2) = \frac{f(x_1,x_2)}{f(x_2)} \tag{A.24}$$

A collection of n random variables $X_1, X_2, \cdots X_n$ is said to be **independent** if the joint probability density function is equal to the product of the marginal probability density functions:

$$p(x_1, x_2, \cdots x_n) = p(x_1)p(x_2) \cdots p(x_n) \tag{A.25}$$

A.4 Expectations

The **expectation** of a random variable, X, is:

$$E(X) = \int_{-\infty}^{\infty} xf(x)dx \tag{A.26}$$

Intuitively, the expectation is an integration (averaging) of x weighted by the probability of x occurring, f(x). It is simply the mean or average value of a random variable, X. For the discrete case:

$$E(X) = \sum_i x_i f(x_i) \tag{A.27}$$

In general:

$$E(X_1 + X_2 + \cdots + X_n) = \\ E(X_1) + E(X_2) + \cdots + E(X_n) \tag{A.28}$$

More specifically, if there are n independent random variables, $X_1, X_2, \cdots X_n$, then [DEGR]:

$$E(\prod_{i=1}^{n} X_i) = \prod_{i=1}^{n} E(X_i) \tag{A.29}$$

This is a consequence of $f(x_1, x_2, \cdots x_n) = f(x_1)f(x_2) \cdots f(x_n)$.

The **nth moment** of a probability density is

$$E(X^n) = \int_{-\infty}^{\infty} x^n f(x)dx \tag{A.30}$$

for the continuous case and

$$E(X^n) = \sum_i x_i{}^n f(x_i) \tag{A.31}$$

for the discrete case. Moments can be used to characterize a probability density function [THOM]. Usually, though, only the lower moments play a significant role. For instance, the **variance** of a random variable, σ^2, is a measure of the "spread" of a density about its mean:

$$\sigma^2 = E[(X-E(X))^2] \tag{A.32}$$

$$\sigma^2 = \int_{-\infty}^{\infty} (x-E(X))^2 f(x)dx \tag{A.33}$$

or for the discrete case:

$$\sigma^2 = \sum_i (x_i-E(x_i))^2 f(x_i) \tag{A.34}$$

The variance is related to the 2nd moment through:

$$\sigma^2 = E(X^2)-(E(X))^2 \tag{A.35}$$

A.5 Convolution

Suppose $Y=X_1+X_2$ where the random variables X_1 and X_2 are independent with marginal densities $f_1(x_1)$ and $f_2(x_2)$. Then [PAPO] [KLEI 75] the density of Y, $f_Y(y)$ is given by:

$$f_Y(y_1) = \int_{-\infty}^{\infty} f_1(y_1-y_2) f_2(y_2) dy_2 \tag{A.36}$$

This function is called the **convolution** of f_1 and f_2. Convolution operations are used in circuit theory, signal processing, and queueing theory.

A.6 Combinatorics

The number of combinations of n elements taken m at a time is:

$$\binom{n}{m} = \frac{n!}{(n-m)!m!} \tag{A.37}$$

References

[ABAT]
Abate, J. and Whitt, W., "Calculating Time-Dependent Performance Measures for the M/M/1 Queue", *IEEE Transactions on Communications*, Vol. 37, No. 10, Oct. 1989, pp. 1102-1104.

[ACKR]
Ackroyd, M.H., "M/M/1 Transient State Occupancy Probabilities Via the Discrete Fourier Transform", *IEEE Transactions on Communications*, Vol. COM-30, No.3, March 1982, pp. 557-559.

A discussion of this technique relative to that of [JONE] appears in [CHIE].

[AMMA]
Ammar, H.H. and Liu, R.W., "Analysis of the Generalized Stochastic Petri Nets by State Aggregation", *International Workshop on Timed Petri Nets*, Torino, Italy, IEEE Computer Society Press, July, 1985, pp. 88-95.

[BAIL]
Bailey, N.T.J., "A Continuous Time Treatment of a Single Queue Using Generating Functions", *Journal of the Royal Statistical Society*, Series B, Vol. 16, pp. 288-291.

[BARD]
Bard, Y., "Some Extensions to Multiclass Queueing Network Analysis", in *Performance of Computer Systems*, ed. by M. Arato, A. Butrimenko and E. Gelenbe, North-Holland, Amsterdam, 1979.

This paper introduced the idea of approximate mean value analysis.

[BASK 75]
Baskett, F., Chandy, K.M., Muntz, R.R. and Palacios, F., "Open, Closed and Mixed Networks of Queues with Different Classes of Customers", *Journal of the ACM*, Vol. 22, No. 2, April 1975, pp. 248-260.

This paper generalizes the earlier work of Jackson, and Gordon and Newell, in describing classes of queueing networks with product form solution. It is often referred to as the "BCMP paper".

[BASK 76]
Baskett, F. and Smith, A.J., "Interference in Multiprocessor Computer Systems with Interleaved Memory", *Communications of the ACM*, Vol. 19, No. 6, June 1976, pp. 327-334.

[BHAN]
Bhandarkar, D.P., "Analysis of Memory Interference in Multiprocessors", *IEEE Transactions on Computers*, Vol. C-24, No. 9, Sept. 1975, pp. 897-908.

[BHAT]
Bhatia, S., Mouftah, H.T. and Ilyas, M., "Hybrid Modeling

Techniques for Complex Computer Networks", *IEEE International Conference on Communications 1987*, Seattle WA., June 1987, pp. 1311-1355.

[BLUM]
Blum, A., et. al., "Experiments with Decomposition of Extended Queueing Network Models", *Modeling Techniques and Tools for Performance Analysis*, D. Potier, Ed., North-Holland, Amsterdam, The Netherlands, 1985 pp. 623-640.

[BODN]
Bodnar, B.L. and Liu, A.C., "Modeling and Performance Analysis of Single-Bus Tightly-Coupled Multiprocessors", *IEEE Transactions on Computers*, Vol. 38, No. 3, March 1989, pp. 464-470.

[BRAU]
Brauer, W., (editor), *Net Theory and Applications*: Proceedings of the Advanced Course on General Net Theory of Processes and Systems, Hamburg, 1979, Springer-Verlag, 1980.

[BREM]
Bremaud, P., *Point Processes and Queues: Martingale Dynamics*, Springer-Verlag, New York, 1981.

[BRUE 78]
Bruell, S.C., *On Single and Multiple Job Class Queueing Network Models of Computing Systems*, Ph.D. Thesis, Computer Sciences Dept., Purdue University, Dec. 1978.

Later condensed into [BRUE 80].

[BRUE 80]
Bruell, S.C. and Balbo, G., *Computational Algorithms for Closed Queueing Networks*, North-Holland, N.Y., 1980.

A very accessible treatment of the convolution algorithm for single and multi-class networks.

[BRUE 86]
Bruell, S.C. and Ghanta, S., "Throughput Bounds for Generalized Stochastic Petri Net Models", *International Workshop on Timed Petri Nets*, Torino, Italy, IEEE Computer Society Press, July 1985, pp. 250-261.

[BRUM]
Brumelle, S.L., "On the Relation Between Customers and Time Averages in Queues", *Journal of Applied Probability*, Vol. 8, No. 3, pp. 508-520.

[BURK]
Burke, P.J., "The Output of a Queueing System", *Operations Research*, Vol. 4, 1956, 699-704.

[BURM]
Burman, D.Y., Lehoczky, , J.P., and Lim, Y., "Insensitivity of Blocking Probabilities in a Circuit-Switching Network", *Journal of Applied*

Probability, Vol. 21, 1984, pp. 850-859.

[BUZE 76]
Buzen, J.P., "Operational Analysis: The Key to the New Generation of Performance Prediction Tools", *Proceedings of IEEE COMPCON*, Sept. 1976.

[BUZE 73]
Buzen, J.P., "Computational Algorithms for Closed Queueing Networks with Exponential Servers", *Communications of the ACM*, Vol. 16, No. 9, Sept. 1973, pp. 527-531.

[CANT]
Cantrell, P.E., "Computation of the Transient M/M/1 Queue cdf, pdf and Mean with Generalized Q-Functions", *IEEE Transactions on Communications*, Vol. COM-34, No. 8, August 1986, pp. 814-817.

[CASS]
Cassandras, C.G. and Strickland, S.G., "Perturbation Analysis Techniques for Communication Networks", *IEEE Global Telecommunications Conference*, New Orleans, LA, Dec. 1985, pp. 13.5.1-13.5.5.

[CHAN 75]
Chandy, K.M., Herzog, U. and Woo, L., "Parametric Analysis of Queueing Networks", *IBM Journal of Research and Development*, Vol. 19, 1975, pp. 43-49.

[CHAN 79]
Chandy, K.M., Holmes, V., and Misra, J., "Distributed Simulation of Networks", *Computer Networks*, Vol. 3, 1979, pp. 105-113.

[CHAN 80]
Chandy, K.M. and Sauer, C.H., "Computational Algorithms for Product Form Queueing Networks", IBM Res. Rep. RC7950, IBM Thomas J. Watson Research Center, Yorktown Heights, NY (1980) (also in the Proceedings Supplement, Performance 80, ACM Press, Toronto, Ont.)

Describes the LBANC algorithm.

[CHAN 81]
Chandy, K.M. and Misra, J., "Asynchronous Distributed Simulation via a Sequence of Parallel Computations", *Communications of the ACM*, Vol. 24, April 1981, pp. 198-206.

[CHAN 82]
Chandy, K.M. and Neuse, D., "Linearizer: A Heuristic Algorithm for Queueing Network Models of Computing Systems", *Communications of the ACM*, Vol. 25, No. 2, 1982, pp. 126-133.

[CHEN]
Chen, W., *Solution Manual for Telecommunications Networks: Protocol, Modeling and Analysis*, Addison-Wesley, 1987.

[CHIE]
Chie, C.M., "Comments on "M/M/1 Transient State Occupancy

Probabilities Via the Discrete Fourier Transform"", *IEEE Transactions on Communications*, Vol. COM-31, No. 2, Feb. 1983, pp. 289-290.

[CHIO]
Chiola, G., "A Software Package for the Analysis of Generalized Stochastic Petri Net Models", *International Workshop on Timed Petri Nets*, July 1985, Torino, Italy, IEEE Computer Society Press, 1986, pp. 136-143.

[CIAR]
Ciardo, G., Muppala, J. and Trivedi, K., "SPNP: Stochastic Petri Net Package", *Proceedings of the Petri Nets and Performance Models*, Kyoto, Japan, Dec. 1989, IEEE Computer Society Press, pp. 142-151.

[COHE]
Cohen, J.W., *The Single Server Queue*, North-Holland Publishing Co., 1969.

[COMF]
Comfort, J.C., "Distributed Simulation on a Network of Transputers", *Proceedings of the European Simulation Multiconf. Simulation Comput. Integrated Manufacturing*, Vienna, Austria, July 1987, pp. 25-29.

[CONW 86]
Conway, A. and Georganas, N. "RECAL - A New Efficient Algorithm for the Exact Analysis of Multiple-Chain Queueing Networks", *Journal of the ACM*, Vol. 33, No. 4, Oct. 1986, pp. 768-791.

A polynomial algorithm for models with multiple routing chains.

[CONW 89]
Conway, A.E., De Souza E Silva, E., and Lavenberg, S.S., "Mean Value Analysis by Chain of Product Form Queueing Networks", *IEEE Transactions on Computers*, Vol. 38, No. 3, March 1989, pp. 432-441.

[CONW 89b]
Conway, A.E. and Georganas, N.D., *Queueing Networks - Exact Computational Algorithm: A Unified Theory Based on Decomposition and Aggregation*, The MIT Press, Cambridge, Mass., 1989.

Includes a discussion of the RECAL algorithm.

[COOP]
Cooper, R.B., *Introduction to Queueing Theory*, North-Holland Publishing Co., New York, 1981.

A solid introduction to classical queueing theory.

[COUR 77]
Courtois, P.-J., *Decomposibility: Queueing and Computer System Applications*, Academic Press, New York, 1977.

[COUR 85]
Courtois, P.-J., "On Time and Space Decomposition of Complex Structures", *Communications of the ACM*, Vol. 28, No. 6, June 1985,

pp. 590-603.

Models of large and complex systems can often be reduced to smaller sub-models through decomposition. This paper discusses certain criteria for successful decompositions.

[COX]
Cox, D.R. and Smith, W.L., *Queues*, Chapman and Hall, London, 1961.

[CRAN]
Crane, M. and Lemoine, A., *An Introduction to the Regenerative Method for Simulation Analysis*, Springer-Verlag, New York, 1977.

[DAVI]
Davidson, D. and Reynolds, P., "Performance Analysis of a Distributed Simulation Algorithm Based on Logical Processes", *Proceedings of the Winter Simulation Symposium*, Arlington, VA, Dec. 1983, pp. 267-268.

[DECE]
Decegama, A., "Parallel Processing Simulation of Large Computer Networks", *Proceedings of the Symposium on Simulation of Computer Networks*, Colorado Springs, CO, Aug. 1987, pp. 51-63.

[DEGR]
DeGroot, M.H., *Probability and Statistics*, Addison-Wesley, Reading, Mass., 1975.

[DENN]
Dennis, J., (editor), *Record of the Project MAC Conference on Concurrent Systems and Parallel Computation*, ACM, New York, June 1970.

[DESO]
De Souza E Silva, E. and Lavenberg, S.S., "Calculating Joint Queue-Length Distributions in Product-Form Queueing Networks", *Journal of the Association of Computing Machinery*, Vol. 36, No. 1, Jan. 1989, pp. 194-207.

[DIAZ]
Diaz, M., "Modeling and Analysis of Communication and Cooperation Protocols Using Petri Net Based Models", *Protocol Verification Specification, Testing and Verification*, C. Sunshine, editor, North-Holland, New York 1982.

[DIDO]
Di Donato, A.R. and Jarnagin, M.P., "A Method for Computing the Circular Coverage Function", *Math. Comput.*, Vol. 16, July 1962, pp. 347-355.

[DIJK]
Dijkstra, E.W., "Cooperating Sequential Processes", *Programming Languages*, F. Genvys, editor, Academic Press, New York, 1968, pp. 43-112.

[DISN]
Disney, R.L., *Traffic Processes in Queueing Networks: A Markov Renewal Approach*, The Johns Hopskins University Press, Baltimore, Md., 1987.

[DOYL]
Doyle, P.G. and Snell, J.L., *Random Walks and Electric Networks*, The Carus Mathematical Monographs No. 22, Published by the Mathematical Association of America, 1984.

A tutorial introduction to the fascinating relation between electric circuits and random walks.

[DUGA]
Dugan, J.B., Ciardo, G., Bobbio, A., and Trivedi, K., "The Design of a Unified Package for the Solution of Stochastic Petri Net Models", *International Workshop on Timed Petri Nets*, Torino, Italy, IEEE Computer Society Press, July 1985, pp. 6-13.

[EAGE]
Eager, D.L., and Lipscomb, J.N., "The AMVA Priority Approximation", *Performance Evaluation*, North-Holland, Vol. 8, No. 3, 1988, pp. 173-193.

The MVA Priority Approximation is proposed as a relatively inexpensive and accurate approximation technique for queueing networks with priority scheduled service centers.

[EILO]
Eilon, S., "A Simpler Proof of $L = \lambda W$", *Operations Research*, Vol. 17, 1969, pp. 915-916.

[EMER]
Emer, J.S., and Ramakrishnan, K., "Performance Considerations for Distributed Services -- A Case Study: Mass Storage", *Proceedings of the 8th International Conference on Distributed Computing Systems*, June 1988, pp. 289-297.

[ERDE]
Erdelyi, A., *Asymptotic Expansions*, Dover Publications, New York, 1956.

See also *Asymptotic Expansions* by E.T. Copson, Cambridge University Press, 1965, Chapters 5,6 and see also *Introduction to Asymptotics and Special Functions* by F.W.J. Oliver, Academic Press, New York, 1974, Chapter 3.

[FELD]
Feldmeier, D.C., "Traffic Measurements on a Token Ring Network", *Proceedings of the Computer Networking Symposium*, Washington D.C., 1986, IEEE Computer Society Press, pp. 236-243.

Describes data traffic measurements made on a ten megabit token ring network at M.I.T. A comparison is made with the Ethernet findings of Shoch and Hupp.

[FELL]
Feller, W., *An Introduction to Probability Theory and Its Applications*, Wiley, New York, 1950.

This is a classic introductory graduate level work on probability theory.

[FILI]Filipiak, J., "Access Protection for Fairness in a Distributed Queue Dual Bus Metropolitan Area Network", *Proceedings of the IEEE International Conference on Communications*, 1989, pp. 635-639.

[FILM]
Filman, R.E. and Friedman, D.P., *Coordinated Computing, Tools and Techniques for Distributed Software*, McGraw-Hill, New York, 1984.

[FIRB]
Firby, P.A. and Gardiner, C.F., *Surface Topology*, published by Ellis Horwood Limited, Chichester and distributed by Halsted Press: a division of John Wiley & Sons, New York, 1982.

A very readable introduction to surface topology.

[FISH 78a]
Fishman, G.S., *Principles of Discrete-Event Simulation*, Wiley, New York 1978.

[FISH 78b]
Fishman, G.S., "Grouping Observations in Digital Simulation", *Management Science*, Vol. 24, 1978, pp. 510-521.

[FLOR]
Florin, G. and Natkin, S., "Evaluation des performances d'un protocole de communication a l'aide des reseaux de Petri et des processus stochastiques", *J. AFCET Mutiprocesseurs et Multiordinateurs en Temps Re'el*, 1978, pp. E0-E26.

This is the first stochastic Petri network paper. A list of early papers by these authors appears in the IEEE Transactions on Computers, Vol. C-32, No. 9, Sept. 1983, pg. 880. Another early, independent, appearance of stochastic Petri nets is in [MOLL 82].

[FREN 87]
Frenkel, K.A., "Profiles in Computing: Alan L. Scherr, Big Blue's Time-Sharing Pioneer", *Communications of the ACM*, Vol. 30, No. 10, Oct. 1987, pp. 824-828.

Scherr discusses his preference for analysis over simulation.

[FROS 86]
Frost, V.S. and Shanmugan, K.S., "Hybrid Approaches to Network Simulation", *IEEE International Conference on Communications 1986*, Toronto, Ontario, June 1986, pp. 228-234.

[FROS 88]
Frost, V.S., Larue, W., and Shanmugan, K.S., "Efficient Techniques for the Simulation of Computer Networks", *IEEE Journal of Selected Areas in Communications*, Vol. 6, No. 1, January 1988, pp. 146-157.

[GARR]
Garrett, M.W. and Li, S.-Q., "A Study of Slot Reuse in Dual Bus Multiple Access Networks", *Proceedings of IEEE INFOCOM'90*, San Francisco, CA, June 1990.

[GERL]
Gerla, M. and Kleinrock, L., "On the Topological Design of Distributed Computer Networks", *IEEE Transactions on Communications*, vol. COM-25, No. 1, Jan. 1977, pp. 48-60.

[GHAN]
Ghanta, S. and Bruell, S.C., "Throughput Bounds for Generalized Stochastic Petri Net Models", *International Workshop on Timed Petri Nets*, Torino, Italy, July 1985, IEEE Computer Society Press, 1986, pp. 250-262.

[GLYN 86a]
Glynn, P. and Sanders, J., "Monte Carlo Optimization of Stochastic Systems: Two New Approaches", *Proceedings of the 1986 ASME Computer Engineering Conference*, 1986.

[GLYN 86b]
Glynn, P., "Optimization of Stochastic Systems", *Proceedings of the Winter Simulation Symposium*, Washington D.C., Dec. 1986, pp. 52-59.

[GORD]
Gordon, W.J. and Newell, G.F., "Closed Queueing Systems with Exponential Servers", *Operations Research*, Vol. 15, 1967, pp. 254-265.

[GREE]
Greenberg, A.G. and McKenna, J., "Solution of Closed, Product Form, Queueing Networks Via The RECAL and Tree-RECAL Methods on a Shared Memory Multiprocessor", *Proceedings of the 1989 ACM SIGMETRICS and Performance'89 International Conference on Measurement and Modeling of Computer Systems*, Berkeley CA, May 1989, pp. 127-135.

[GRES]
Gressier, E. "A Stochastic Petri Net Model for Ethernet", *International Workshop on Timed Petri Nets*, Torino, Italy, July 1985, IEEE Computer Society Press, 1986, pp. 296-306.

[GROS]
Gross, D. and Harris, C.M., *Fundamentals of Queueing Theory*, Wiley, New York, 1974, 1985.

A very thorough and excellent introduction to classical queueing theory.

[HAAS]
Haas, P.J. and Shedler, G.S., "Regenerative Simulation of Stochastic Petri Nets", *International Workshop on Timed Petri Nets*, Torino, Italy, IEEE Computer Society Press, July 1985, pp. 14-21.

[HAMI 86a]
Hamilton, Jr., R.L., *Modeling and Analysis of Multihop and Priority Random Access Computer Networks*, Ph.D Thesis, School of Electrical Engineering, Purdue University, 1986.

[HAMI 86b]
Hamilton, Jr., R.L. and Coyle, E.J., "A Two-Hop Packet Radio Network with Product Form Solution", *Proceedings of the 1986 Information Sciences and Systems Conference*, Princeton University, Princeton N.J., March 1986, pp. 871-876.

[HAMI 89]
Hamilton, Jr., R.L. and Yu, C., "A Buffered Two Node Packet Radio Network with Product Form Solution", *Proceedings of IEEE INFOCOM'89*, Ottawa, Canada, April 1989, pp. 520-528.

[HAMM]
Hammond, J.L. and O'Reilly, P.J.P., *Performance Analysis of Local Computer Networks*, Addison-Wesley, Reading, Mass., 1986.

A self-contained and highly readable treatment of this topic.

[HEID 81]
Heidelberger, P. and Welch, P.D., "A Spectral Method for Confidence Interval Generation and Run Length Control in Simulation", *Communications of the ACM*, Vol. 24, 1981, pp. 233-245.

[HEID 84]
Heidelberger, P., and Lavenberg, S.S., "Computer Performance Evaluation Methodology", *IEEE Transactions on Computers*, Vol. C-33, Dec. 1984, pp. 1195-1220.

[HEID 85]
Heidelberger, P., "Statistical Analysis of Parallel Simulation", *Proceedings of the Winter Simulation Symposium*, Washington D.C., Dec. 1985, pp. 290-295; also in IBM Corp., Yorktown Heights, NY, Tech. Rep. RC12733.

[HEID 86]
Heidelberger, P., "Limitations of Infinitesimal Perturbation Analysis", IBM Thomas J. Watson Research Center, Yorktown Heights, NY, *Tech. Rep.*, RC11891, May 1986.

[HERZ]
Herzog, U., Woo, L. and Chandy, K.M., "Solution of Queueing Problems by a Recursive Technique", *IBM Journal of Research and Development*, May 1975, pp. 295-300.

This is the first paper to systematically look at recursive solutions for non-product form models.

[HEYM]
Heyman, D.P. and Stidham Jr., S., "The Relation Between Customer and Time Averages in Queues", *Operations Research*, Vol. 28, No. 4.

[HO 83a]
Ho, Y.C. and Cassandras, C., "A New Approach to the Analysis of Discrete Event Dynamic Systems", *Automatica*, Vol. 19, 1983, pp. 149-167.

[HO 83b]
Ho, Y.C. and Cao, X., "Perturbation Analysis and Optimization of Queueing Networks", *Journal of Optimization Theory and Applications*, Vol. 40, 1983, pp. 554-582.

[HO 84]
Ho, Y.C., et. al., "Perturbation Analysis and Optimization of Large Multiclass (Non-Product Form) Queueing Networks", *Large Scale Systems*, Vol. 7, 1984, pp. 165-180.

[HOLT]
Holt, A., Saint, H., Shapiro, R. and Warshall, S., Final Report of the Information System Theory Project, *Technical Report RADC-TR-68-305*, Rome Air Development Center, Griffiss Air Force Base, New York, 1968, 352 pages.

[HOWA]
Howard, R.A., *Dynamic Probabilistic Systems: Vol. I: Markov Models*, John Wiley & Sons, New York, 1971.

[HUI]Hui, J.Y. and Arthurs, E., "A Broadband Packet Switch for Integrated Transport", *IEEE Journal on Selected Areas in Communications*, Vol. SAC-5, No. 8, Oct. 1987, pp. 1264-1273.

This paper was originally presented at the International Conference on Communications, ICC'87, at Seattle WA, June 1987.

[HWAN]
Hwang, K. and Briggs, F.A., *Computer Architecture and Parallel Processing*, McGraw-Hill, New York, 1984.

An up to date treatment.

[JACK 57]
Jackson, J.R., "Networks of Waiting Lines", *Operations Research*, Vol. 5, 1957, pp. 518-521.

The first product form solution paper.

[JACK 64]
Jackson, J.R., "Job Shop Like Queueing Systems", *Management Sciences*, Vol. 10, No. 1, 1964, pp. 131-142.

Generalizes the work of [JACK 57].

[JAIN]
Jain, R. and Turner, R., "Workload Characterization Using Image Accounting", *Proceedings of the Computer Performance Evaluation Users Group 18th Meeting*, Oct. 1982, pp. 111-120.

[JEFF]
Jefferson, D., "Virtual Time", *ACM Transactions on Programming*

Languages and Systems, Vol. 7, July 1985, pp. 404-425.

[JENQ]
Jenq, Y.-C., "Performance Analysis of a Packet Switch Based on Single-Buffered Banyan Network", *IEEE Journal on Selected Areas in Communication*, Vol. SAC-1, No. 6, Dec. 1983, pp. 1014-1021.

[JEWE]
Jewell, W.S., "A Simple Proof of $L = \lambda W$", *Operations Research*, Vol. 15, 1967, pp. 1109-1116.

[JONE]
Jones, S.K., Cavin III, R.K. and Johnston, D.A., "An Efficient Computational Procedure for the Evaluation of the M/M/1 Transient State Occupancy Probabilities", *IEEE Transactions on Communications*, Vol. COM-28, No. 12, Dec. 1980, pp. 2019-2020.

For a discussion of the merits of this technique relative to that in [ACKR] see [CHIE].

[KARO 86]
Karol, M.J., Hluchyj, M.G. and Morgan, S.P., "Input Vs. Output Queueing on a Space-Division Packet Switch", *IEEE Global Communications Conference*, Houston TX, Dec. 1986, pp. 659-665.

[KARO 87]
Karol, M.J., Hluchyj, M.G. and Morgan, S.P., "Input Vs. Output Queueing on a Space-Division Packet Switch", *IEEE Transactions on Communications*, Vol. COM-35, No. 12, Dec. 1987, pp. 1347-1356.

[KARO 88]
Hluchyj, M.G. and Karol, M.G., "Queueing in Space-Division Packet Switching", *IEEE INFOCOM '88*, New Orleans La., April 1988, pp. 334-343.

[KAUF]
Kaufman, L., Gopinath, B., and Wunderlich, E.F., "Analysis of Packet Network Congestion Control Using Sparse Matrix Algorithms", *IEEE Transactions on Communications*, Vol. COM-29, No. 4, April 1981, pp. 453-465.

[KELL]
Kelly, F.P., *Reversibility and Stochastic Networks*, John Wiley & Sons, New York, 1979.

[KELL 75]
Kelly, F.P., "Networks of Queues with Customers of Different Types", *Journal of Applied Probability*, Vol. 12, 1975, pp. 542-554.

[KELL 76]
Kelly, F.P., "Networks of Queues", *Advances in Applied Probability*, Vol. 8, 1976, pp. 416-432.

A very good introduction to the concept of reversibility. A great many interesting and unusual stochastic models are presented.

[KEND]
Kendall, D.G., "Some Problems in the Theory of Queues", *Journal of the Royal Statistical Society*, Series B, Vol. 13, No. 2, 1951, pp. 151-185.

The simple imbedded Markov chain proof of the Pollaczek-Khinchin mean value formula used in this book first appears in this paper. Contains an extensive discussion section.

[KHIN]
Khinchin, A.Y., "Mathematisches uber die Erwartung vor einem offentlichen Schalter" or "Mathematical Theory of Stationary Queues", *Matem. Sbornik*, Vol. 39, 1932, pp. 73-84.

This paper is one of two early analysis of the M/G/1 queueing system. It is in Russian with a German summary. See also [POLL] and [KEND].

[KIM]Kim, H.S. and Leon-Garcia, A., "Performance of Buffered Banyan Networks Under Nonuniform Traffic Patterns", *IEEE INFOCOM'88*, New Orleans, La., April 1988, pp. 344-353.

[KLEI 64]
Kleinrock, L., *Communication Nets*, McGraw-Hill, New York, 1964, Reprinted: Dover, New York, 1972.

[KLEI 75]
Kleinrock, L., *Queueing Systems, Vol. I: Theory*, Wiley, New York, 1975.

A very thorough and excellent introduction to classical queueing theory. Vol. II deals with applications.

[KOBA]
Kobayashi, H., *Modeling and Analysis: An Introduction to System Performance Evaluation*, Addison-Wesley, Reading, Mass. 1978.

[KOGA]
Kogan, Ya.A. and Boguslavsky, L.B., "Asymptotic Analysis of Memory Interference in Multiprocessors with Private Cache Memories", *Performance Evaluation*, Vol. 5, 1985, pp. 97-104.

[KRIT]
Kritzinger, P.S., "A Performance Model of the OSI Communication Architecture", *IEEE Transactions on Communications*, Vol. COM-34, No. 6, 1986, pp. 554-563.

[KURO]
Kurose, J.F. and Mouftah, H.T., "Computer-Aided Modeling, Analysis, and Design of Communication Networks", *IEEE Journal on Selected Areas in Communications*, Vol. 6, No. 1, Jan. 1988, pp. 130-145.

This is the paper in which Chapter 4's discussion on simulation first appeared.

[LAVE 80]
Lavenberg, S.S. and Reiser, M., "Stationary State Probabilities of Arrival Instants for Closed Queueing Networks with Multiple Types of Customers", *Journal of Applied Probability*, Vol. 17, 1980, pp. 1048-1061.

This article presents the *Arrival Theorem* and deals with a generalization of BCMP type networks.

[LAVE 83]
Lavenberg, S.S. (ed.), *Computer Performance Modeling Handbook*, Academic Press, New York, 1983.

Includes a particularly strong summary of analytical queueing results and extended discussions of simulation.

[LAW 82]
Law, A.M. and Kelton, W.D., *Simulation Modeling and Analysis*, McGraw-Hill Book Co., New York, 1982.

A solid introductory text.

[LAW 83]
Law, A., "Statistical Analysis of Simulation Output Data", *Operations Research*, Vol. 31, Nov. 1983, pp. 983-1029.

[LAZA 84a]
Lazar, A.A. and Robertazzi, T.G., "The Geometry of Lattices for Markovian Queueing Networks", *Columbia University Tech. Rep., 1984, SUNY at Stony Brook CEAS Tech. Rep. 471.*.

[LAZA 84b]
Lazar, A.A. and Robertazzi, T.G., "The Geometry of Lattices for Multiclass Markovian Queueing Networks", *Proceedings of the 1984 Information Sciences and Systems Conference*, Princeton University, Princeton N.J., March 1984, pp. 164-168.

[LAZA 84c]
Lazar, A.A., "An Algebraic Topological Approach to Markovian Queueing Networks", *Proceedings of the 1984 Information Sciences and Systems Conference*, Princeton University, Princeton N.J., March 1984, pp. 437-442.

[LAZA 86]
Lazar, A.A. and Robertazzi, T.G., "The Algebraic and Geometric Structure of Markovian Petri Network Lattices", *Proceedings of the Twenty-Fourth Annual Allerton Conference on Communication, Control and Computing*, University of Illinois, Urbana-Champaign Ill., 1986, pp. 834-843.

[LAZA 87]
Lazar, A.A. and Ferrandiz, J.M., "Geometric Analysis of Qusai-Birth and Death Processes by Flow Redirection", *Proceedings of the 1987 Conference on Information Sciences and Systems*, The Johns Hopkins University, Baltimore, MD, March 1987, pp. 865-870.

[LAZA 87b]
Lazar, A.A. and Robertazzi, T.G., "Markovian Petri Net Protocols with Product Form Solution", *IEEE INFOCOM'87*, San Francisco, CA, 1987, pp. 1054-1062.

[LAZO]
Lazowska, E., Zahorjan, J., Cheriton, D., and Zwaenepoel, W., "File Access Performance of Diskless Workstations", *ACM Transactions on Computer Systems*, Vol. 4, No. 3, Aug. 1986.

[LITT]
Little, J.D.C., "A Proof of the Queueing Formula $L = \lambda W$", *Operations Research*, Vol. 9, 1961, pp. pp. 383-387.

[LUBA]
Lubachevsky, B., and Ramakrishnan, K., "Parallel Time-Driven Simulation of a Network using Shared Memory MIMD Computer", *Modeling Tools and Techniques for Performance Analysis*, D. Potier, Ed., North-Holland, Amsterdam, 1985.

[MARC]
Marcum, J.I., "A Statistical Theory of Target Detection by Pulsed Radar", *IRE Transactions on Information Theory*, Vol. IT-6, pp. 59-267, April 1960.

[MARS]
Marsan, M. Ajmone, Balbo, G., Conte G., *Performance Models of Multiprocessor Systems*, The MIT Press, Cambridge, Mass., 1986.

An excellent treatment of modeling multiprocessors using queueing and Petri Nets.

[MARS 83]
Marsan, M.J., Balbo, G., Conte, G. and Gregoretti, F., "Modeling Bus Contention and Memory Interference in a Multiprocessor System", *IEEE Transactions on Computers*, Vol. C-32, No. 1, 1983, pp. 60-72.

Presents the original multiprocessor models upon which the modified product form models of chapter 5 are based.

[MARS 84]
Marsan, M., Balbo, G. and Conte G., "A Class of Generalized Stochastic Petri Nets for the Performance of Evaluation of Multiprocessor Systems", *ACM Transactions on Computer Systems*, May 1984.

Introduces immediate transitions for stochastic Petri Nets.

[MARS 85]
Marsan, M., Balbo, G., Bobbio, A., Chiola, G., Conte, G., and Cumani, A., "On Petri Nets with Stochastic Timing", *International Workshop on Timed Petri Nets*, Torino, Italy, IEEE Computer Society Press, July 1985, pp. 80-87.

[MASS]
Massey, W., *Algebraic Topology: An Introduction*, Springer-Verlag, New York, 1977.

[McKEN 81]
McKenna, J., Mitra, D., and Ramakrishnan, K.G., "A Class of Closed Markovian Queueing Networks: Integral Representations, Asymptotic Expansions, and Generalizations", *The Bell System Technical Journal*, Vol. 60, No. 5, May-June 1981, pp. 599-641.

[McKEN 82]
McKenna, J. and Mitra, D., "Integral Representations and Asymptotic Expansions for Closed Markovian Queueing Networks: Normal Usage", *The Bell System Technical Journal*, Vol. 61, No. 5, May-June 1982, pp. 661-683.

[McKEN 84]
McKenna, J. and Mitra, D., "Asymptotic Expansions and Integral Representations of Moments of Queue Lengths in Closed Markovian Networks", *Journal of the ACM*, Vol. 31, No. 2, April 1984, pp. 346-360.

[McKEN 86]
McKenna, J. and Mitra, D., "Asymptotic Expansions for Closed Markovian Networks with State-Dependent Service Rates", *Journal of the ACM*, Vol. 33, No. 3, July 1986, pp. 568-592.

[McKEN 87]
McKenna, J., "Asymptotic Expansions of the Sojourn Time Distribution Functions of Jobs in Closed, Product-Form Queueing Networks", *Journal of the ACM*, Vol. 34, No. 4, Oct. 1987, pp. 985-1003.

[MISR]
Misra, J., "Distributed Discrete Event Simulation", *ACM Computing Surveys*, Vol. 18, No. 1, March 1986, pp. 39-66.

[MIYA]
Miyazawa, M., "Time and Customer Processes in Queues with Stationary Inputs", *Journal of Applied Probability*, Vol. 14, No. 2, pp. 349-357.

[MOLL]
Molloy, M.K., *Fundamentals of Performance Modeling*, Macmillan Publishing Co., New York, 1989.

A very good introduction into the basics of performance evaluation theory.

[MOLL 82]
Molloy, M.K., "Performance Analysis Using Stochastic Petri Nets", *IEEE Transactions on Computers*, Vol. C-31, No. 9, 1982, pp. 913-917.

[MOLL 84]
Molloy, M.K., "Modeling and Analysis of LAN Protocols Using Labeled Petri Nets", *Technical Report 84-15*, Dept. of Computer Science, The University of Texas at Austin, Austin, TX, 1984.

[MOLL 86]
Molloy, M.K., "Fast Bounds for Stochastic Petri Nets", *International*

Workshop on Timed Petri Nets, Torino, Italy, IEEE Computer Society Press, July 1985, pp. 244-249.

[NEUT]
Neuts, M.F., *Matrix-Geometric Solutions in Stochastic Models: An Algorithmic Approach*, The Johns Hopkins University Press, Baltimore Md., 1981.

[NEWE]
Newell, G.F., *Applications of Queueing Theory*, Chapman and Hall, London, 1971.

[NEWM]
Newman, R.M. and Hullet, J.L., "Distributed Queueing: A Fast and Efficient Packet Access Protocol for QPSX", *Proceedings of the International Conference on Computer Communications* (ICCC), 1986.

[NICH]
Nichols, W.G. and Emer, J.S., "Design and Implementation of the VAX Distributed File Service", *Digital Technical Journal*, No. 9, June 1989, pp. 16-28.

[OREI]
O'Reilly, P., and Hammond, J.L., "An Efficient Technique for Performance Studies of CSMA/CD Local Networks", *IEEE Journal of Selected Areas in Communications*, Vol. SAC-2, Jan. 1984, pp. 238-249.

[OUST]
Ousterhout, J., et. al., "A Trace-driven Analysis of the UNIX 4.2 BSD File System", *Proceedings of the 10th ACM Symposium on Operating Systems Principles*, Dec. 1985.

[PAPO]
Papoulis, A., *Probability, Random Variables and Stochastic Processes*, McGraw-Hill, New York, 1965.

[PARE]
Pareek, S. and Robertazzi, T.G., "An Algorithm for the Exact Decomposition of a Class of Non-Product Form Queueing Models", *Proceedings of the 1990 Conference on Information Sciences and Systems*, Princeton N.J., March 1990.

[PARZ]
Parzen, E., *Stochastic Processes*, Holden-Day, San Francisco, 1962.

[PEAC]
Peacock, J., Wong, J., and Manning, E., "Distributed Simulation using a Network of Processors", *Computer Networks*, Vol. 3, 1970, pp. 44-56.

[PERR 84]
Perros, H.G., "Queueing Networks with Blocking: A Bibliography", *Performance Evaluation Review*, Vol. 12, Aug. 1984, pp. 8-12.

[PERR 89]
Perros, H.G. and Altiok, T., ed., *Proceedings of the First International Workshop on Queueing Networks with Blocking*, North Carolina State University, Raleigh N.C., May 1988, Elsevier North-Holland, 1989.

[PETE]
Peterson, J.L., *Petri Net Theory and the Modeling of Systems*, Prentice Hall, Englewood Cliffs N.J., 1981.

An excellent introductory text on Petri Nets.

[PETR]
Petri, C.A., *Kommunikation mit Automaten*, Ph.D Thesis, Univ. of Bonn, Fed. Rep. of Germany, 1962.

This is the original work on Petri Nets.

[POLL]
Pollaczek, F., "Uber eine Aufgabe dev Wahrscheinlichkeitstheorie", *Math. Zeitschrift*, Vol. 32, 1930, pp. 64-100, 729-750.

One of the two early analysis of the M/G/1 queueing system. See also [KHIN] and [KEND].

[RAMA]
Ramakrishnan, K.G. and Mitra, D., "An Overview of PANACEA, a Software Package for Analyzing Markovian Queueing Networks", *The Bell System Technical Journal*, Vol. 61, No. 10, Dec. 1982, pg. 2849.

[RAMA 86]
Ramakrishnan, K. and Emer, J., "A Model of File Server Performance for a Heterogeneous Distributed System", *Proceedings of the ACM SIGCOMM '86 Symposium*, August 1986, pp. 338-347.

[RAMA 89]
Ramakrishnan, K. and Emer, J., "Performance Analysis of Mass Storage Service Alternatives for Distributed Systems", *IEEE Transactions on Software Engineering*, vol. 15, no. 2, 1989.

[REED]
Reed, D., Malony, A., and McCredie, B., "Parallel Discrete Simulation: A Shared Memory Approach", *Performance Evaluation Review*, Vol. 15, May 1987, pp. 36-39.

[REIB]
Reibman, A., Smith, R. and Trivedi, K., "Markov and markov reward model transient analysis: An overview of numerical approaches", *European Journal of Operational Research*, Vol. 40, 1989, North-Holland, pp. 257-267.

[REIC]
Reich, E., "Waiting Times when Queues are in Tandem", *Annals of Mathematical Statistics*, Vol. 28, 1957, pp. 768-773.

[REIM]
Reiman, M., and Weiss, A., "Sensitivity Analysis via Likelihood Ratios", *Proceedings of the Winter Simulation Symposium*,

Washington D.C., Dec. 1986, pp. 285-289; extended version available as unpublished report.

[REIS 73]
Reiser, M. and Kobayashi, H., "Recursive Algorithms for General Queueing Networks with Exponential Servers", *IBM Research Report RC 4254*, Yorktown Heights N.Y., 1973.

[REIS 75]
Reiser, M. and Kobayashi, H., "Queueing Networks with Multiple Closed Chains: Theory and Computational Algorithms", *IBM Journal of Research and Development*, Vol. 19, 1975, pp. 283-294.

[REIS 79]
Reiser, M., "A Queueing Network Analysis of Computer Communications Networks with Window Flow Control", *IEEE Transactions on Communications*, Vol. COM-27, No. 8, August 1979, pp. 1199-1209.

[REIS 80]
Reiser, M. and Lavenberg, S.S., "Mean-value Analysis of Closed Multichain Queueing Networks", *Journal of the ACM*, Vol. 27, No. 2, April 1980, pp. 313-322.

[REIS 81]
Reiser, M., "Mean-value Analysis and Convolution Method for Queue-Dependent Servers in Closed Queueing Networks", *Performance Evaluation*, Vol. 1, 1981, pp. 7-18.

The discussion here relates MVA to the convolution algorithm and to the LBANC algorithm.

[REIS 82]
Reiser, M., "Performance Evaluation of Data Communication Systems", *Proc. IEEE*, vol. 70, no. 2, Feb. 1982, pp. 171-195.

A wide ranging survey paper.

[ROBE 88a]
Robertazzi, T.G. and Schwartz, S.C., "Best 'Ordering' for Floating Point Addition", *ACM Transactions on Mathematical Software*, Vol. 14, No. 1, March 1988, pp. 101-110.

An error analysis of various schemes for accurately adding large numbers of positive floating point numbers.

[ROBE 89]
Robertazzi, T.G., "Recursive Solution of a Class of Non-Product Form Protocol Models", *IEEE INFOCOM'89*, Ottawa, Canada, April 1989, pp. 38-46.

[RUBI]
Rubinstein, R.Y., *Simulation and the Monte Carlo Method*, Wiley, New York, 1981.

A thorough mathematical treatment of Monte Carlo simulation.

[SAAT]
Saaty, T.L., *Elements of Queueing Theory*, McGraw-Hill, New York, 1961.

[SAND]
Sanders, W.H. and Meyer, J.F., "METASAN: A Performability Evaluation Tool Based On Stochastic Activity Networks", *Proceedings of the ACM-IEEE Comp. Soc. Fall Joint Comp. Conf.*, Nov. 1986.

[SAUE 76]
Sauer, C.H., et.al., "Hybrid Analysis/Simulation of Distributed Networks", IBM Thomas J. Watson Research Center, Yorktown Heights NY, *Research Report* RC 6341, 1976.

[SAUE 81]
Sauer, C. and Chandy, K.M., *Computer System Performance Modeling*, Prentice-Hall, Englewood Cliffs NJ, 1981.

[SAUE 83]
Sauer, C.H. and MacNair, E.A., *Simulation of Computer Communication Systems", Prentice-Hall, Englewood Cliffs, N.J., 1983.*

A readable treatment which makes use of the RESQ queueing software package.

[SCHO]
Schoemaker, Ed., *Computer Networks and Simulation II and III*, North-Holland, Amsterdam, The Netherlands, 1982.

[SCHR]
Schruben, L., "Detecting Initialization Bias in Simulation Output", *Operations Research*, Vol. 30, 1982, pp. 569-590.

[SCHW 66]
Schwartz, M., Bennett, W.R. and Stein, S., *Communications Systems and Techniques*, McGraw-Hill, New York, 1966.

[SCHW 82]
Schwartz, M., "Performance Analysis of the SNA Virtual Route Pacing Control", *IEEE Transactions on Communications*, Vol. COM-30, Jan. 1982, pp. 172-184.

[SCHW 87]
Schwartz, M., *Telecommunication Networks: Protocols, Modeling and Analysis*, Addison-Wesley, Reading, Mass., 1987.

An up to date and comprehensive treatment. Also a good introduction to commonly used analytical techniques.

[SCHWE]
Schweitzer, P., "Approximate Analysis of Multiclass Closed Networks of Queues", *Proc. of the International Conference on Stochastic Control and Optimization*, Amsterdam, 1979.

[SHAN]
Shanthikumar, J. and Sargent, R., "A Unifying View of Hybrid Simulation/Analytical Models and Modeling", *Operations Research*,

Vol. 31, Nov.-Dec. 1983, pp. 1030-1052.

[SHAP]
Shapiro, S.D., "A Stochastic Petri Net with Applications to Modeling Occupancy Times for Concurrent Task Systems", *Networks*, Vol. 9, 1979, pp. 375-379.

[SHED]
Shedler, G.S. and Haas, P.J., "Regenerative Simulation of Stochastic Petri Nets", *International Workshop on Timed Petri Nets*, Torino, Italy, July 1985, IEEE Computer Society Press, 1986, pp. 14-21.

[STEP]
Stephenson, M. and Molloy, M.K., "Measurements of Terminal Character Input Behavior", *Proceedings of the Computer Networking Symposium*, Washington D.C. 1986, IEEE Computer Society Press, pp. 227-235.

This paper describes the results of an experimental study of character interarrival times from asynchronous terminals attached to a time sharing system in a university environment.

[STID]
Stidham Jr., S., "A Last Word on $L = \lambda W$", *Operations Research*, Vol. 22, 1974, pp. 417-421.

[SURI 83]
Suri, R., "Robustness of Queueing Network Formulas", *Journal of the Association of Computing Machinery*, Vol. 30, No. 3, July 1983, pp. 564-594.

[SURI 84]
Suri, R., "Perturbation Analysis Gives Strongly Consistent Estimates for the M/G/1 Queue", Div. of Applied Science, Harvard University, Cambridge, MA, *Tech. Rep.*, 1984.

[SURI 87]
Suri, R., "Infinitesimal Perturbation Analysis of Discrete Event Dynamic Systems", *Journal of the ACM*, Vol. 34, July 1987, pp. 686-717.

[TAKA]
Takacs, L., *Introduction to the Theory of Queues*, Oxford University Press, New York, 1962.

[THOM]
Thomas, J.B., *An Introduction to Statistical Communication Theory*, Wiley, New York, 1969.

A very readable introductory work.

[TURN]
Turner, J., "Design of a Broadcast Packet Network", *IEEE INFOCOM 86*, Miami, FL, April 1986, pp. 667-675.

[VAND]
van Dijk, N.M., "A Simple Bounding Methodology for Non-Product Form Finite Capacity Queueing Systems", *Proceedings of the First International Workshop on Queueing Networks with Blocking*, North Carolina State University, May 1988, Elsevier North-Holland, H.G. Perros and T. Altiok, eds., 1989.

[VANS]
Van Slyke, R., Chou, W. and Frank H., "Avoiding Simulation in Simulating Communication Networks", *Proceedings of the 1972 National Computer Conference*, pp. 165-169.

[WAGN]
Wagner, D.B. and Lazowska, E.D., "Parallel Simulation of Queueing Networks: Limitations and Potentials", *Proceedings of the 1989 ACM SIGMETRICS and PERFORMANCE'89 International Conference on Measurement and Modeling of Computer Systems*, Berkeley, California, May 1989, pp. 146-155.

[WALR]
Walrand, J., "Quick Simulation of Queueing Networks", *Proceedings of the 2nd International Workshop on Applied Mathematics and Performance/Reliability Models of Comput./Commun. Systems*, Rome,Italy, May 1987, pp. 275-286.

[WALR 88]
Walrand, J., *An Introduction to Queueing Networks*, Prentice-Hall, Englewood Cliffs N.J., 1988.

[WANG 86]
Wang, I.Y. and Robertazzi, T.G., "The Probability Flux Circulation of Certain Communication and Computation State Models", *Proceedings of the 1986 Information Sciences and Systems Conference*, Princeton University, Princeton N.J., March 1986, pp. 865-870.

[WANG]
Wang, I.Y. and Robertazzi, T.G., "Recursive Computation of Steady State Probabilities of Non-Product Form Queueing Networks Associated With Computer Network Models", *IEEE Transactions on Communications*, Vol. 38, No. 1, Jan. 1990, pp. 115-117.

[WANG 89]
Wang, I.Y. and Robertazzi, T.G., "Service Stage Petri Net Protocols with Product Form Solution", *Proceedings of the 1989 ACM Sigmetrics and Performance '89 International Conference on Measurement and Modeling of Computer Systems*, Berkeley, CA, May 1989, pg. 233. An extended version appears in SUNY at Stony Brook, College of Engineering and Applied Science Technical Report 534, Dec. 29, 1988, available from authors upon request.

[WELC]
Welch, P., "The Statistical Analysis of Simulation Results", in *The Performance Modeling Handbook*, S. Lavenberg, Ed., Academic Press,

New York, 1983.

[WIES]
Wieselthier, J.E. and Ephremides, A., "Some Markov Chain Problems in the Evaluation of Multiple-Access Protocols", *Proceedings of The First International Conference on the Numerical Solution of Markov Chains*, sponsored by N.C. State University Computer Science Dept., N.C. State University, Raleigh, N.C., Jan. 1990, pp. 258-282.

[WONG]
Wong, J., Moura, J. and Field J., "Hierarchical Modeling of Local Area Computer Networks", *Proceedings of the IEEE National Telecommunications Conference*, Houston, Texas, Dec. 1980, pp. 37.1.1-37.1.5.

[WONG CY]
Wong, C.Y., Dillon, T.S. and Forward, K.E., "Timed Places Petri Nets with Stochastic Representation of Place Time", *International Workshop on Timed Petri Nets*, Torino, Italy, IEEE Computer Society Press, July 1985, pp. 96-103.

[WU]Wu, C.-l. and Feng, T.-y., *Tutorial: Interconnection Networks for Parallel and Distributed Processing*, IEEE Computer Society Press, 1984.

A large collection of previously published papers on interconnection networks.

[YANG]
Yang, T., Posner, M.J.M. and Templeton, J.G.C., "A Generalized Recursive Technique for Finite Markov Processes", *Proceedings of The First International Conference on the Numerical Solution of Markov Chains*, sponsored by N.C. State University Computer Science Dept., N.C. State University, Raleigh, N.C., Jan. 1990, pp. 216-237.

[ZAHO]
Zahorjan, J., Eager, D.L., and Sweillam, H.M., "Accuracy, Speed and Convergence of Approximate Mean Value Analysis", *Performance Evaluation*, North-Holland, Vol. 8, No. 4, August 1988, pp. 255-270.

A new modification of the approximate Mean Value Analysis algorithm for load independent networks is presented. The convergence of AMVA techniques is also examined.

[ZAZA]
Zazanis, M. and Suri, R., "Estimating First and Second Derivatives of Response Time for G/G/1 Queues from a Single Sample Path", Div. of Applied Science, Harvard University, *Tech. Rep.*, 1985.

[ZENI]
Zenie, A., "Colored Stochastic Petri Nets", *International Workshop on Timed Petri Nets*, July 1985, Torino, Italy, IEEE Computer Society Press, 1986, pp. 262-271.

Index