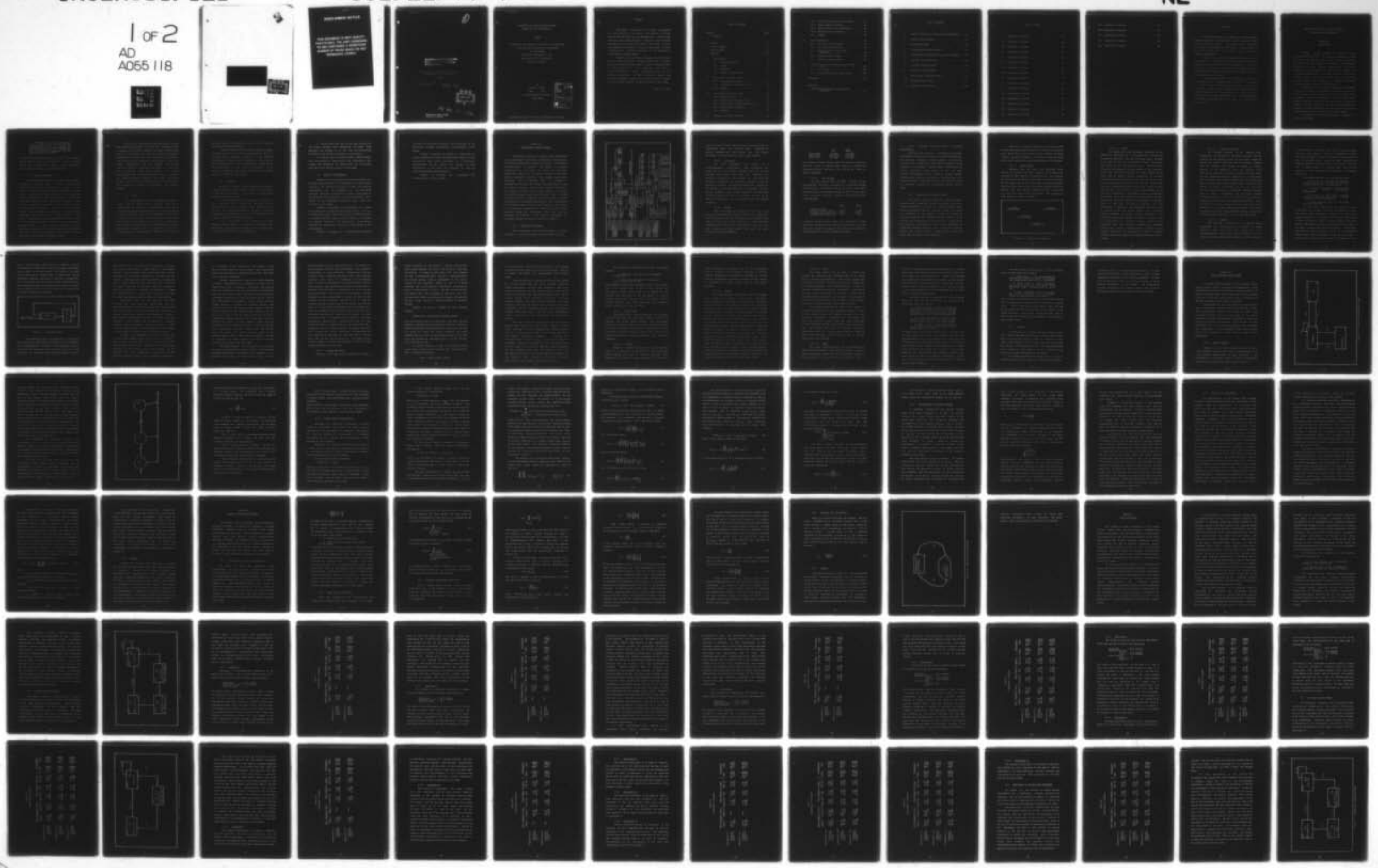
AD-A055 118 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO SCH--ETC F/G 9/2
VALIDATION OF CLOSED QUEUEING NETWORK MODELS FOR THE DECSYSTEM---ETC(U)
MAR 78 H E SAXTON

UNCLASSIFIED GCS/EE/77-7 NL

1 OF 2
AD
A055 118



GCS/EE/78-7

(1)

# VALIDATION OF CLOSED QUEUEING NETWORK MODELS FOR THE DECSYSTEM-10

THESIS

GCS/EE/78-7 ✓

Harold E. Saxton
Captain USAF

GCS/EE/78-7

# VALIDATION OF CLOSED QUEUEING NETWORK MODELS FOR THE DECSYSTEM-10

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science

by

Harold E. Saxton

Capt USAF

Graduate Computer Science

March 1978

## Preface

This report is the result of a study to determine the applicability of the product form solution of Closed Queueing Network Models for The DECsystem-10. I believe that this work will enable personnel of the Air Force Avionics Laboratory to more effectively utilize this class of model for performance evaluation. I hope that this study will lay a foundation for continued improvement of the DECsystem-10 Closed Queueing Network Model.

I would like to thank the personnel of the Air Force Avionics Laboratory, whose patience permitted me to gather the data necessary for this report. I would especially like to thank Major Bob E. Baker, AFAL/AAF-2, and James C. McCool and Michael E. Price, software specialists at the Avionics Laboratory, for their dedicated support and assistance. Finally I would like to thank Dr. Gary B. Lamont for his continued support and advice throughout this study.

Harold E. Saxton

## Table of Contents

## List of Figures

## List of Tables

## Abstract

A validation study was performed on Closed Queueing Network Models for the DECsystem-10 located at the Air Force Avionics Laboratory (AFAL). The purpose of this study was to provide the personnel of AFAL with an understanding of the accuracy with which this class of model can predict computer performance.

A previously developed Fortran program was used to implement the product form solution and the computational algorithms required for performance evaluation. A review of the solution technique is presented with a discussion of the consequences resulting from assumptions required to arrive at the closed form solution.

The validation results are described for a series of experiments which investigated the accuracy of the models for several system configurations and under a variety of workloads. These results indicate that Closed Queueing Network Models are a flexible means of representing the DECsystem-10 and that the product form solution provides performance predictions which are useful for evaluation of the system.

# VALIDATION OF CLOSED QUEUEING NETWORK MODELS FOR THE DECSYSTEM-10

# Chapter I

## Introduction

Computer systems continue to exert an ever increasing influence over all aspects of management in the Air Force. Over the last two decades, computer structures and operation have changed dramatically. The increased capability of modern computer systems has often been obtained by the introduction of more complex systems. The increased computer system complexity makes it difficult to understand which factors most directly affect the operation of the system. The process of computer performance evaluation seeks to identify these factors.

The first step of a computer performance evaluation (CPE) is to identify the measures of performance which are most important. These measures may include response time, throughput, resource utilization, cost, and many others which are general in nature or meaningful only for specific modes of computer operation. When the desired performance measures have been identified, the next step is to select a technique which can be applied to the desired system.

One approach to computer performance evaluation is the use of modeling.

"Modeling is the process of mapping a real system into a suitable abstract representation, such as equation or queueing diagram. It is the modeler who decides those aspects of a system that are sufficiently important to be represented (Ref 20;309)."

In order to provide usable numerical results, some method of solution must be found for the model. For example, solution methods for queueing models might include simulation, approximation techniques or algebraic methods.

## I.1 Problem Statement

The DECsystem-10 owned by the Air Force Avionics Laboratory is operated under a workload which varies considerably throughout the workday. The system is a dual processor system with multiprogramming capability. It is not readily apparent from conventional high level operational summaries produced by the system what the effects on performance measures will be when there are changes in the workload or apparent limitations of system capacity caused by high priority customers. The use of Closed Queueing Network Models provides one method of evaluating system performance under the various conditions found at the Avionics Laboratory. A Fortran program which solves a class of Closed Queueing Network Models has been developed by McKenzie (Ref 1). The program produces selected performance measures for Closed Queueing Network Models after the parameters and structure of the model have been specified by the computer performance analyst.

The class of Closed Queueing Network Models has not been adequately validated for use on the DECsystem-10. The validation of this class of model is necessary before these models can reasonably be used for computer performance evaluation of a system such as the DECsystem-10. In order to develop some level of confidence in this type of model, the system representation must be specified and various workloads applied to determine the amount of error. The ideal validation study would analyze all system configurations and all workloads which could be placed on the system. Validation studies of this degree are seldom possible and the validation effort generally investigates a small portion of possible configurations and workload changes.

## I.2 Scope

The representation of the DECsystem-10 using a Closed Queueing Network Model requires that system configuration and workload be specified. The selection of system configuration and workload for this validation effort is greatly restricted due to time limitations. This report is directed at two common configurations of the CPU: dual processor and single processor. The workloads considered in this report are not necessarily representative of actual system users. However, they are adaptable to represent almost any set of resource demands. The use of a variety of workloads permits more detailed error analysis than was possible with previous studies.

In order to better understand the validation effort, a closed form solution of Closed Queueing Network Models is presented with emphasis placed on the physical interpretation of many of the assumptions of the solution. The validation effort itself covers multiple workloads for each configuration and also considers adjustments for the solution to compensate for assumptions which are not totally satisfied in the real world system.

## I.3 Approach

In order to validate closed queueing network models, a thorough knowledge of the computer system and assumptions of the model is desirable. The following procedures were performed during this study:

1. The DECsystem-10 was analyzed to determine the major components and to relate the operation of these components to the class of model being investigated. In addition, the available performance measuring tools were investigated to determine the nature of the data which could be used to validate a performance model.

2. The development of the solution for closed queueing network models was analyzed to determine the restrictions which would be required for direct application of this class of model to the DECsystem-10.

3. Modifications were made to the data structure of the Fortran program which implemented the model. This permitted the program to be run from the time-sharing terminals at the Air Force Institute of Technology.

4. Several forms of Closed Queueing Network Models were investigated for application to the DECsystem-10. The results were summarized in tabular form and an analysis performed on the observed errors of each model.

## I.4 Order of Presentation

Chapter II of this report describes the hardware and software characteristics of the DECsystem-10 at the Avionics Laboratory. The hardware description includes the physical configuration and the parameters of the individual resources. Software characteristics include a description of the Operating System resource allocation procedures and a review of the performance evaluation tools which are available on the system.

Chapter III presents the development of a closed form solution for closed queueing network models. The class of models to be used in this report allows different classes of customers and four types of service center to represent the computer resources. The implications of assumptions required for solution are analyzed as they relate to the DECsystem-10.

Chapter IV presents the computationally efficient algorithms for performance measures. The limitations on the application of these algorithms is also presented in the chapter.

Chapter V presents the results of application of various structures of closed queueing network models to the DECsystem-10. Each model corresponds to a particular representation of the system. The results of each configuration of the model are discussed as well as comparisons between representations.

Chapter VI presents the conclusions and recommendations for future study.

# Chapter II

# DECsystem-10 Computer System

Quantitative measures of computer system performance are only meaningful when system configuration is specified. The system configuration includes all hardware and software characteristics. In general, the number of possible configurations is very large, even though the performance effects are usually due to a small fraction of the total number of parameters (Ref 5;4). The computer performance evaluation work by McKenzie (Ref 1) includes a description of hardware and software characteristics of the DECsystem-10 located at the Air Force Avionics Laboratory. The addition of 512K words of core memory is the only significant hardware change since McKenzie's work. In order to avoid excessive references to earlier works, a brief description of the hardware configuration will be presented with emphasis placed on those areas relevent to this study. A more detailed presentation of selected portions of the Operating System and performance tools will also be presented to establish a more complete background for DECsystem-10 model validation discussed in Chapter V.

## II.1 DECsystem-10 Hardware

The configuration of the AFAL DECsystem-10 is shown in Figure 1. The dual processor system utilizes two KI10

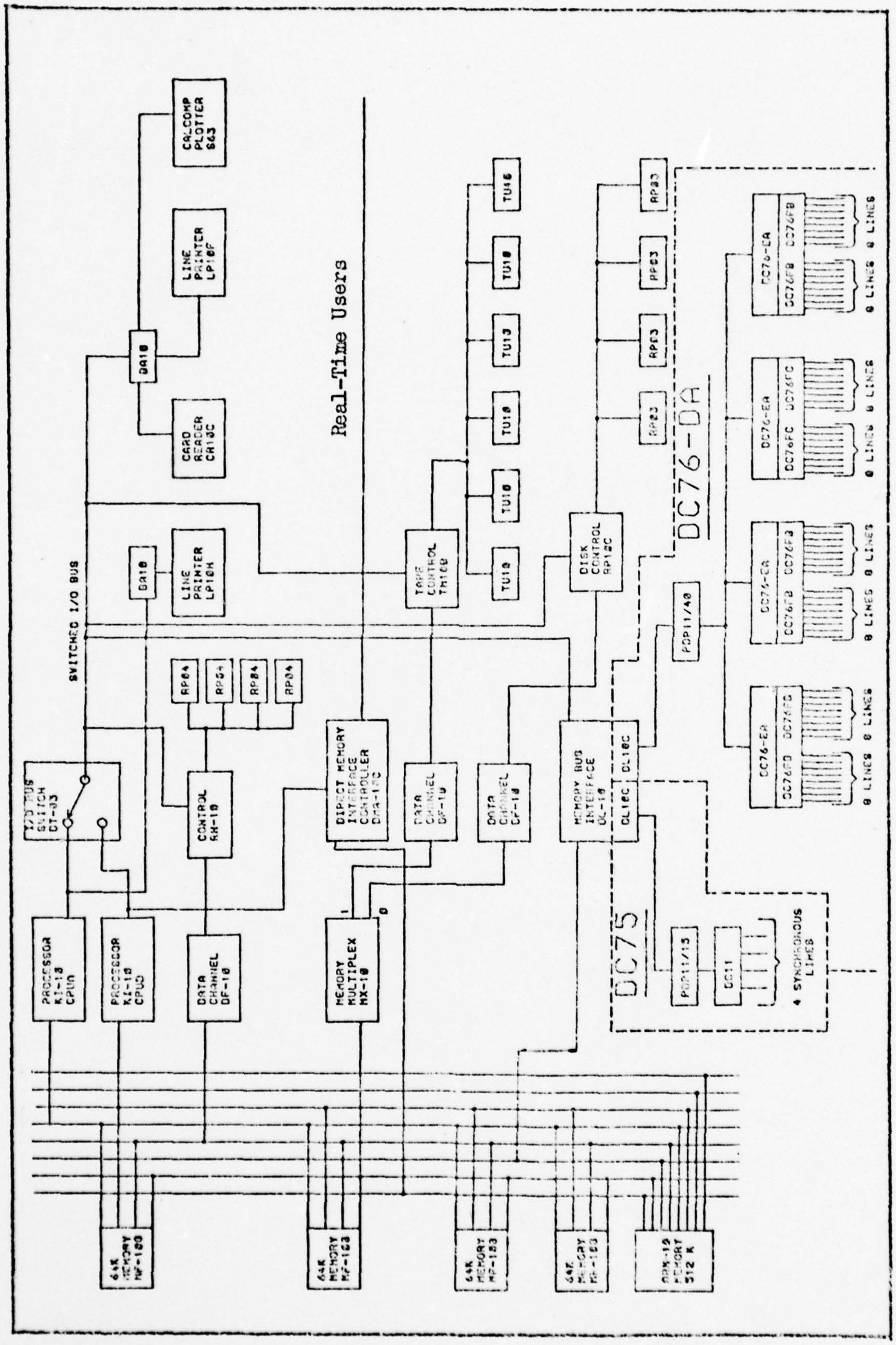


Figure 1. Avionics Laboratory DECsystem-10 Configuration

central processor units, which provide processing service to timesharing, batch, and real-time users. Components of special interest for this report are: the central processors, the core memory and mass storage devices.

### II.1.1 Processors

The KI10 processors are joined in a primary/secondary configuration. Both processors have access to all core memory. However, only the primary processor may perform the normal I/O functions. The secondary processor may only perform I/O associated with real-time processing. The secondary processor does not perform any resource allocation. Therefore the majority of the Operating System overhead will be associated with the primary processor. When a job running on the secondary processor requires I/O, the job is stopped, returned to the queue, and identified for execution by only the primary processor.

### II.1.2 Memory

Since the study by McKenzie (Ref 1), the core storage has been expanded with the addition of 512K words. The main memory of the AFAL DECsystem-10 now consists of two types of memory units; four 64K units of Digital Equipment Corporation (DEC) memory (Ref 2;70C-384-01c) and one 512K unit of Ampex Corporation memory (Ref 3;3). The units have the following parameters:

|  | DEC | AMPEX |
|---|---|---|
| Word Length | 36 bits | 36 bits |
| Memory Size | 512K words | 512K words |
| Cycle Time | 1000 nsec | 920 nsec |

The addition of faster memory causes the required processing time for a job to decrease. The amount of decrease is dependent upon the structure of the job and the number of memory references.

### II.1.3 Mass Storage

Two disk systems, RP03 and RP04, are used by the DECsystem-10. Four RP03 disk units are interfaced through the MX10 memory multiplexor channel, while four high speed RP04 disk units are connected through an RH-10 control unit. These disk units have the following characteristics (Ref 2;70C-384-01g):

|  | RP03 | RP04 |
|---|---|---|
| Number of Units | 4 | 4 |
| Capacity (Million Words) | 10.24 | 20.00 |
| Transfer Rate (words/sec) | 66,667 | 178,571 |
| Average Access Time (sec) | 47.5 | 36.8 |

Of special concern to this study is the system use of each disk type. One RP03 unit is used for system files, while one RP04 unit is used for swapping. Under heavy load conditions, interference may result for either disk type.

### II.1.4 Real-Time Processing Effects on Hardware Configuration

McKenzie's work provides a comprehensive discussion of problems arising from real-time programming applications. The excessive swapping overhead induced by real-time program's use of memory has been nearly eliminated by the additional core storage. However, future applications may require increasingly large amounts of core and again restrict time-sharing user's access to memory. Another consequence of real-time processing is the allocation of a single processor to time-sharing users. The dedicated use of the secondary processor is investigated further in this study.

## II.2 DECsystem-10 Operating System

The TOPS-10 operating system, or MONITOR, performs the accounting, scheduling, resource allocation, and service routines necessary to operate in a multiprogramming, time-sharing environment (Ref 4). Numerous tables and queues are maintained by the system in order to perform the required functions. The MONITOR maintains control of all processing by responding to a clock interrupt generated for each processor. The clocks are run at line frequency. The interrupts therefore bracket a time slice 1/60 second in length. This measure of time, called a jiffy, will prove useful when discussing system operation.

There are a variety of modules and programs within the Operating System which permit it to perform its required tasks. Among these are the Control Cycle modules, including resource allocation modules, and various evaluation tools which enable system performance measurements to be obtained.

### II.2.1 Control Cycle

Beginning with the clock tick interrupt, every sixtieth of a second, the cyclic routines of the MONITOR are executed and then control is returned to the user program. The activity of the control cycle takes on added importance for this study. One of the primary data gathering programs, METER, utilizes code located in strategic modules within the MONITOR. The major MONITOR modules activated during each clock tick are: CLOCK1, COMCON and SCHED1. The sequence of these modules is shown in Figure 2.

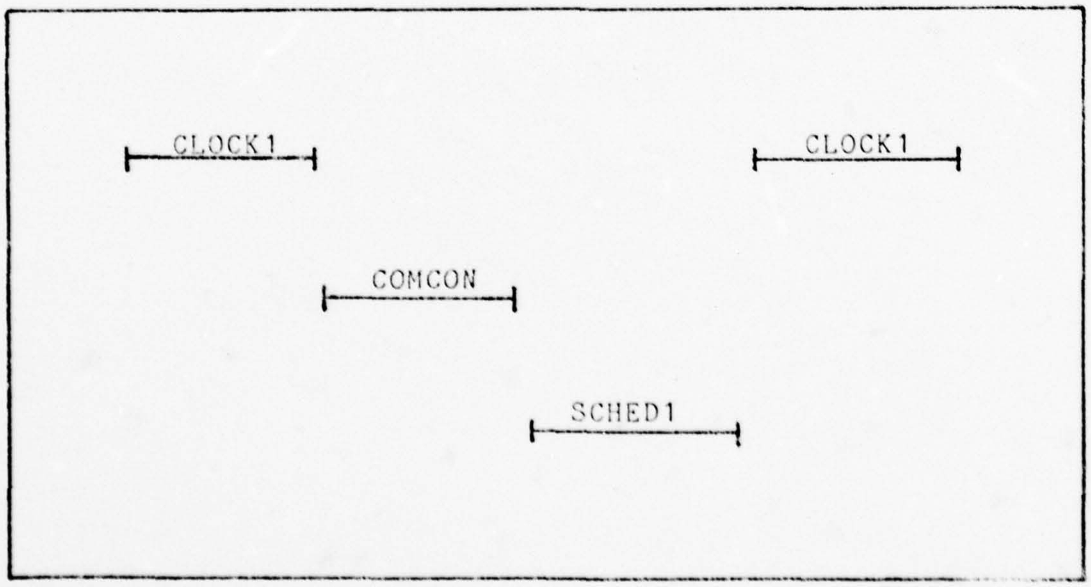


Figure 2. Control Cycle Sequence

#### II.2.1.1 Clock1

In CLOCK1, the basic accounting functions of the MONITOR are performed. If a user program was running, the accumulated time since the last accounting update is added to the job's run total in the MONITOR's Process Data Block. When there are no jobs ready to run, a Monitor routine called the Null program is given control of the processor. If the Null program was running, the time is added to the total Null job time in the CPU Data Block. If the Null program was running and there was at least one job in any of the run queues which could not be run for some reason, then the time is considered "Lost" CPU time. One situation where Lost time may occur is when a job has been swapped out to disk and is scheduled to run before it can be swapped back into memory. If no jobs were in a run queue while the Null program was processed, the time is considered "Idle" time (Ref 4). This time is used to calculate the CPU utilization. In addition to the accounting function, CLOCK1 is entered at the end of the control cycle to perform context switching if needed. Context switching is the process of saving user register contents and pointers. This is required only if a new job has been selected by the scheduler. When the accounting functions are complete, CLOCK1 calls a routine which determines if there are any commands waiting to be processed. If there are commands waiting, control is passed to the command processor routine, COMCON.

### II.2.1.2 Command Processor

The Command Processor is the MONITOR module responsible for interpreting the user's request from the terminal and then passing control to the proper routine to satisfy the command. After the user has entered a command from the terminal, the command is stored in a buffer internal to the MONITOR. If there are commands in the buffer, then COMCON will activate routines which will gather the data necessary for later reference to the appropriate user's terminal. The command processor will not process more than one command during each control cycle. If more than one command is waiting, the command processor will be called during successive cycles to process the remaining commands (Ref 4). This is consistant with the basic MONITOR philosophy which is to run the control cycle quickly to completion and allow the user programs the majority of the processing time. Some commands may require more time than is desirable to be taken during the control cycle. These commands are set up as special MONITOR routines, and are run during the user's time slice as one of the user's jobs.

### II.2.1.3 SCHED1

The scheduler, SCHED1, is the MONITOR module responsible for controlling the allocation of system resources among user jobs. The DECsystem-10 is a multiprogramming system which permits numerous users to reside in core with concurrent operation. It is the function of the scheduler to determine which job will run during each time slice. When swapping occurs, as is permitted at AFAL, then jobs may reside in core or on secondary storage devices such as the RP04 disks. Because of this situation, the scheduler must not only decide which job will run, but also call a Swapper routine to determine which jobs will be swapped to or from secondary storage and when this swapping will occur.

The overall objectives for the scheduler are:

1) Provide for sharing the machine's resources among the user jobs so that each receives his share on a timely basis.

2) Provide fast response capability for jobs that require conversational interaction.

3) Provide a very fast response capability for jobs performing tasks that require "real time" interaction.

4) Make efficient use of all system resources (Ref 7;1).

To meet these objectives, the scheduler uses a modification of the Round Robin (RR) algorithm, one of the most widely used scheduling algorithms for time-shared computer systems. The structure for the Round Robin system is shown in Figure 3. Newly arriving customers take their place at the back of the queue and progress to the head of the queue in first-come-first-serve fashion (Ref 8;166). When a job is ready to receive service, it is allocated a service limit called a quantum. If the job still requires service when the quantum has expired, it is returned to the end of the same queue and the cycle is repeated. The job will continue to alternately wait in the queue and receive service until its service demands are satisfied, at which time it will move to another queue. A special case of the Round-Robin queueing discipline is Processor Sharing (PS). For a PS resource, the service time quantum approaches zero. Each customer receives an infinitely small amount of service before returning to the end of the queue. The customers are then equally sharing the resource.

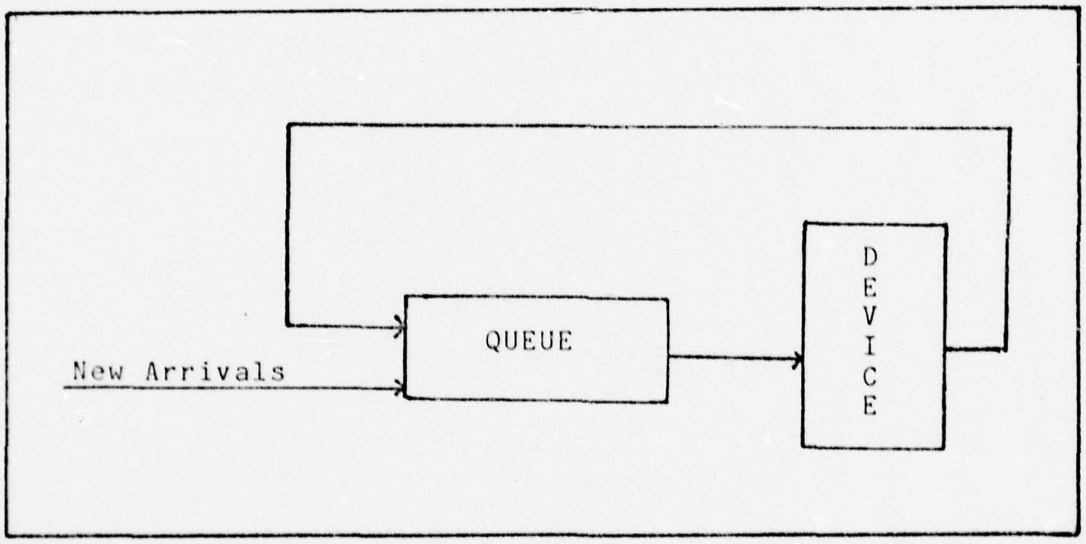


Figure 3. Round-Robin Queue

The Round-Robin queue, designated PQ2, is especially well suited for a time-sharing environment as is encountered on the DECsystem-10. It permits good turnaround time for small jobs even though very large jobs may be on the system (Ref 7;2). It also gives each job in succession an equal opportunity to use the resource associated with the queue. Each of the jobs receive a "fair share" of the resource capacity. In this instance, "fair share" means that no job, regardless of its service requirements, may take over the resource. Different users may have differing opinions on what is a "fair share" for their particular jobs. Some jobs may require much less service than would be allocated to them in the Round-Robin algorithm. For these jobs it might be considered unfair to require them to wait in the queue for large time quantum which they will not fully utilize.

Interactive jobs are one class of job which generally have small processing requirements. The Text Editor and Corrector (TECO) is an example of an interactive job. For this type job, a fast response time capability is desired. When the user enters a TECO command, he expects a quick response, (preferably less than 5 seconds). The time required to cycle through the PQ2 queue will be highly dependent upon the number of jobs in the queue. If the system is heavily loaded, the time required to cycle through PQ2 could easily be several times the desired response.

To meet this requirement for faster response time for small jobs, another queue, PQ1, is used. This queue will have a higher priority than PQ2. A time quantum is also established for jobs in PQ1. However, the PQ1 quantum will generally be much smaller than that received in PQ2 (Ref 7;2). If a job requires more service after its PQ1 quantum has expired, then it will be requeued to the back of the PQ2 queue. In this manner, all jobs receive a small burst of service while in the PQ1 queue. Jobs which still require service enter the Round Robin cycle for PQ2, where they remain until completed.

Certain programs may require even faster response and better performance than is obtained by using PQ1. To satisfy this class of programs, additional high priority queues are established. They are labeled HPQ1 through HPQ15 (Ref 7;6). The Line Printer Spooler and certain computer performance evaluation programs are examples of the type of job which require these queues. If not in use, the Line Printer Spooler is swapped to secondary storage. When printing is required, the spooler must be swapped into core ahead of other user jobs and must have access to the CPU ahead of user jobs so that the output buffer may be filled. Performance measuring programs often require rapid access to MONITOR data tables. If high priority queues were not available, the CPE programs would have to wait in the PQ1 queue and possibly lose data. Jobs in the High Priority queues may acquire all system resources ahead of user programs. However, those jobs which are permitted into HPQ are generally I/O bound jobs so that most of the CPU capacity is available to the user jobs.

The final objective of the scheduler is to operate the system as efficiently as possible within the previously mentioned performance constraints. One approach is to balance the CPU and I/O bound jobs in core so that multiprogramming is made most efficient by overlapping I/O with processing. When the primary/secondary, dual processor configuration is used, additional considerations must be included in the scheduling philosophy. Since the secondary processor, CPU1, has no I/O capability for time-sharing users, it would be desirable to process CPU bound jobs on that processor. That approach would permit I/O bound jobs or jobs with small processing requirements to be run on the primary processor. This approach is incorporated into the scheduler so that consideration is given the type of job assigned to each processor. Jobs to be run on CPU0 are selected in order from the front of the higest priority run queues as follows: HPQ1 through HPQ15, PQ1, and finally PQ2. Using the assumption that jobs which have already run for a relatively long period of time will be CPU bound, the jobs to be run on CPU1 are selected from the front of the lower priority run queues in the following order: PQ2, PQ1, and finally the HPQ's. Another factor is the realization that a compromise must be achieved between throughput and short term response. One of the controlling considerations for this compromise is the number of jobs in the short time run queue, PQ1, and the long run queue, PQ2. The number of jobs in each queue is directly dependent upon the size of the quantum allocation in each queue.

### II.2.2 <u>Quantum Run Time</u>

The flow of jobs through the processor run queues is greatly affected by the amount of service they receive before being requeued and also by the length of time they are permitted to remain in main memory. The job's quantum run time is decremented while the job is being processed. The in-core protect time is decremented if the job is scanned by the scheduler, whether or not it is being serviced. The in-core protect time affects processing of jobs in another manner as well. When the in-core protect time expires, the job is requeued to the end of PQ2. This may occur well before its alloted time quantum has expired, and even if the job is not actually swapped to secondary storage.

Quantum run time is computed by the following formula:

QUANTUM RUN = MIN[QMX,QAD+(SIZE*QML)/QRANG]

Size is the job's size in K (1024 words). QMX, QAD, QML and QRANG are Operating System parameters which are stored in MONITOR tables with values specified for each queue and with the option to change the values, depending upon which swapping device is used. For the synthetic jobs used in this study, the quantum run allocations are: 3 jiffies for PQ1, and 30 jiffies for PQ2.

A similar formula is used to determine in-core protect time. In this case the Minimum Core Usage Function, MCUF, is defined as follows:

MCUF = (PROT * SIZE) + PROTO

PROT and PROTO are internal Monitor parameters. The current parameters at AFAL produce an in-core protect time value of 2 seconds, 120 jiffies, for the synthetic jobs of this study.

The determination of quantum run time for each queue and in-core protect time can have a profound effect on system performance. For PQ1, the quantum run time is a measure of the time the job receives rapid service after entering the system. When this quantum expires, the job is requeued to the end of PQ2 where it receives a new run quantum. However, the job's in-core protect has been decreased. This procedure permits small fast jobs to receive exceptional service scheduling response and also reduces swapping since the longer jobs are still permitted to receive some service in PQ2 before they are swapped out (Ref 4).

For jobs in PQ2, the quantum run time and in-core protect time control the bias of the scheduler for I/O versus CPU jobs. If both parameters are increased, there will be a decrease in the number of jobs whose time slice has expired. Accordingly, swapping will decrease and throughput should improve (because of reduced overhead in swapping). Response is degraded in this instance because jobs will tend to wait longer to swap into core. Decreasing the two parameters will have the opposite effect on throughput and response. The ratio of the two parameters is also important. Longer quantum run times favor CPU jobs, while longer in-core protect times favor I/O jobs.

Two additional restrictions are placed on swapping (Ref 4):

1) Real-time jobs may not be swapped out of core.

2) Jobs with incomplete I/O request may not be swapped out of core.

The scheduler calls up the swapping routine which selects jobs to be swapped into or out of core, within the restrictions stated above. Jobs to be swapped in are selected from the front of the run queues in order of priority: HPQ's, PQ1, and PQ2. Jobs to be swapped out are selected from the end of the run queues from lowest to highest priority: PQ2, PQ1, and HPQ's.

### II.2.3 System Tools

Three performance measuring programs are available on the DECsystem-10. They are SYSTAT, TRACK, and METER. In addition, the SCRIPT program permits the simultaneous execution of several copies of a single program. Execution of various combinations of programs along with one of the performance measuring programs allows the collection of detailed information on system performance under controlled conditions.

#### II.2.3.1 SCRIPT

SCRIPT is a system program which permits numerous copies of a program to be executed. For this study, two basic programs were used as synthetic workload. Using SCRIPT, up to 14 copies of each program can be logged onto pseudo teletypes. In addition, one of the performance measuring programs can be initiated each time so that the resulting performance data would be a measure for a known workload. Simultaneous execution of several "scripts" permit any combination of jobs; limited only by the system's capacity.

### II.2.3.2 Systat

The SYSTAT program provides the user or computer center manager with general system status information. Before logging onto the system, a user may use the SYSTAT command to determine the number of jobs on the system and the percent Null time (idle time plus lost time). Limited information about each user is also available. The SYSTAT program provides each user's job number, program-project number, and input device. Also available is the name and size of the program each user is running and the accumulated run time since logging onto the system. Another measure of system load which is provided by SYSTAT is a listing of active devices and amount of free storage for each disk structure. Of special interest to the system manager is the amount of swapping space used, virtual core used, swapping ratio, virtual core saved by sharing, and average job size (Ref 10). SYSTAT does provide useful information under actual load conditions. However, a more flexible program is needed to gather the data needed for model validation. The desired flexibility is provided by the TRACK program.

### II.2.3.3 Track

The TRACK program may be used to monitor the progress and performance of individual jobs or it may be used to measure the performance of the entire system. For this study, the TRACK program was used only to gather data on system performance. The program performs a "peek" at selected performance parameters at intervals specified by the user. TRACK is initiated by entering a command string which specifies requirements such as: type of report, use of high priority queue, lock in core, length of peek interval, and number of intervals used in each report (Ref 9).

The report generated by TRACK will include a frequency distribution for most parameters measured as well as the average value and standard deviation for each parameter. Information concerning CPU performance was especially important to this study. The TRACK information for each processor includes: percent Null time, percent Lost time, and percent time spent for overhead. Additional measurements useful for model validation are the number of jobs in the run queues, and the number of jobs in I/O wait queues other than teletype.

### II.2.3.4 Meter

The meter facility permits the implementation of "meter points" throughout the MONITOR, wherever a value or event of potential interest may occur. The Meter facility is used to select and collect performance statistics from the

MONITOR. It allows privileged user-mode programs to collect system data for performance analysis or tuning of jobs or of the system. The Meter facility is capable of providing simple statistics or raw data. The raw data outputs require processing and formatting after the program has been run. However, data in this form provides a more detailed picture of the system operation and is more desirable for the early stages of CPE model development.

The Meter facility is comprised of three mechanisms which are used for the collection and disposition of performance data (Ref 9;14.1):

1) Meter Points are short sections of code (often only two instructions) located throughout the MONITOR. This code test for activation of its particular section of code and if active, calls a Meter Point Routine.

2) Meter Point Routines are short, fast routines which process the meter values before passing control to the Meter Channel.

3) Meter Channels are fast routines which dispose of the processed values into a buffer under user control.

The buffer used to accumulate metered data is located in the metering job's core image. The metering job uses the Hibernate monitor call to synchronize the accumulation and processing of data. The buffer length is set to an integral power of two so that indexing may be performed by a simple modulo operation. The MONITOR does not guard against buffer overflow, but does provide the Hibernate function. When large amounts of data are required, the metering job may have to be placed in a high priority queue in order to prevent buffer writeover.

The data gathering routines used in this validation study included three meter points:

> 1) Meter point 1, Job State Queueing. This point is located in SCHED1 at the point where destination queue has been determined.
>
> 2) Meter point 2, Context Switching. This point is located in CLOCK1 where it has determined that a different job is to be run.
>
> 3) Meter point 500, Wait State Code. This point is located in SCHED1 where a change in state code has been required.

When raw data is collected, it must be received and buffered as it is generated. In order to avoid losing data, a routine which is locked in memory must be provided so that the data may be dumped onto secondary storage. If the routine is not locked in memory, it may not be swapped back into core in time to make the transfer when the system is experiencing a high swapping rate.

## II.3 Summary

The DECsystem-10 is a dual processor computer system with a complex Operating System which permits time-sharing, batch and real-time users to share the resources of the system. The complexity of the system arises not so much from the hardware configuration as from the Operating System which attempts to both optimize resource performance and provide some degree of equality to the users.

The system measuring tools provide various levels of information which may be useful for performance evaluation.

A desirable goal of a performance model is that it require only readily available system information in order to make acceptable predictions of the system performance. In order to fully develop the model, it is often necessary to obtain detailed information of the system. The DECsystem-10 provides evaluation tools which present both levels of information which greatly aided this validation effort.

# Chapter III

# Closed Queueing Network Model

The paper by McKenzie (Ref 1) used several Closed Queueing Network Models to evaluate the performance of the DECsystem-10 at the Air Force Avionics Laboratory (AFAL). In order to validate models of this type under various workload conditions, a thorough understanding of the assumptions and solution techniques of the model is desirable.

The solution of the Closed Queueing Network Models used in this report is based on the work of Baskett, Chandy, Muntz, and Palacios (Ref 11). This class of model is an extension of earlier works by Jackson (Ref 12) and Gordon and Newell (Ref 13). The solution technique of Gordon and Newell is presented in Appendix A. In this chapter the work of Baskett, et. al. is reviewed to provide the background for a discussion of the assumptions which most directly affect the application of this class of model to the DECsystem-10.

## III.1 Work of Baskett

Baskett, et. al. have extended earlier works to allow different classes of customers. A representation of a Closed Queueing Network Model is shown in Figure 4. The introduction of multiple customer classes overcomes the limitation of previous works which required that all

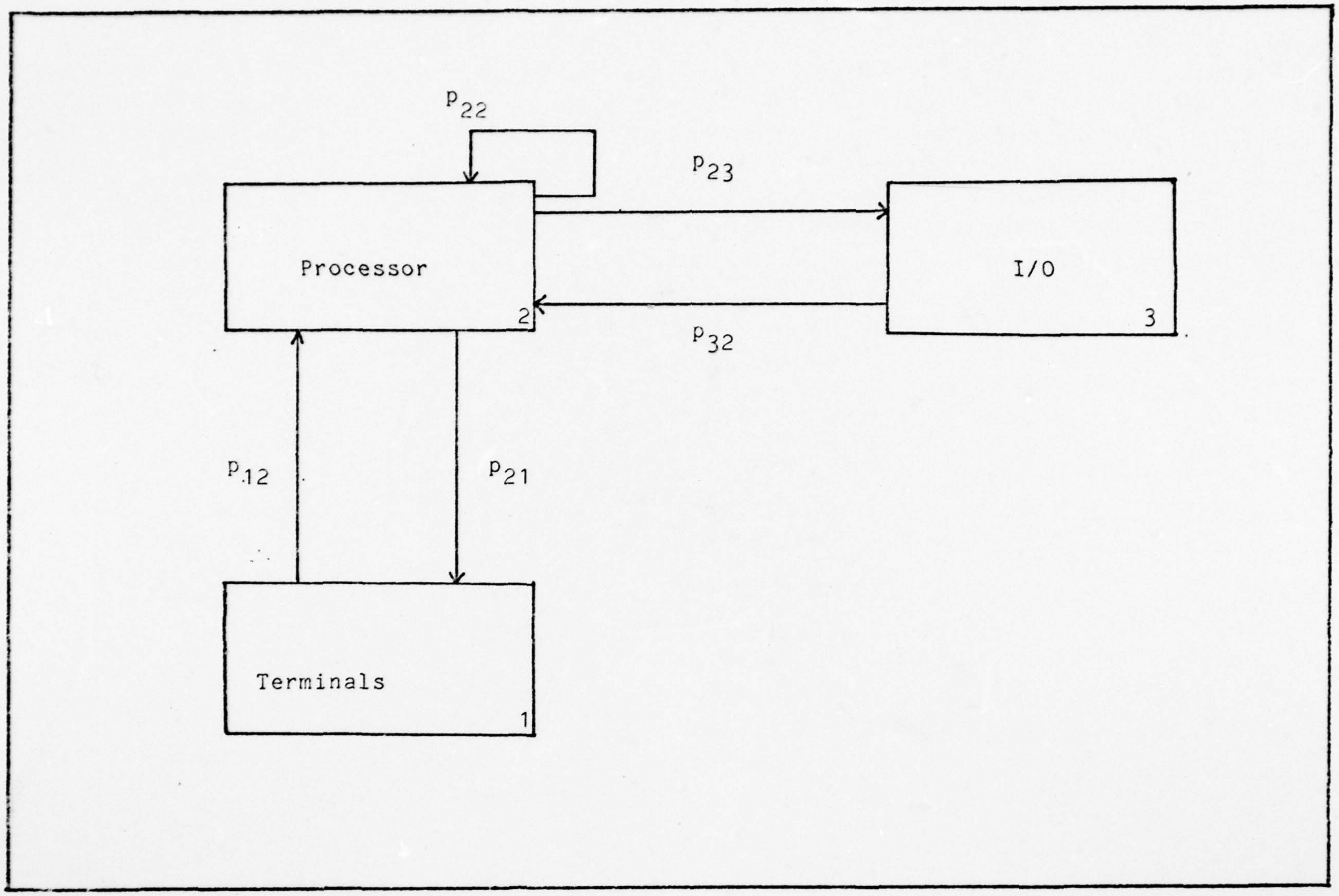


Figure 4. Closed Queueing Network Model

customers belong to the same class. Customers receive service at the resources in the network according to an exponential service time distribution specified for each node in the network. Mean service time for class r customers at node i is given by $1/\mu_{ir}$. Movement of customers between resources is specified by a fixed transition matrix $P=[p_{ir;js}]$ where $p_{ir;js}$ is the probability that a class r customer leaving the ith service center will change to a class s customer and go to the jth service center. Customers within a given class are described by the same service time distribution and the same transition probabilities. The emphasis of this paper is towards validation of the model for customers which do not change class. Later computational consideration will expressly prohibit class changes. This simply means that $p_{ir;js}=0$ when $r\neq s$.

The assumption of exponential service time distributions is not valid for all resources which are likely to appear in a computer system model. Actual system measurements on the Michigan Terminal System indicated that the service times for that system's swapping drum were hyperexponential (Ref 14;147).

Cox (Ref 15) has shown that any service time distribution with rational Laplace Transform may be represented as a series of exponential distributions. The customers in service at a service center which is represented in this fashion are specified by the stage of service obtained. Figure 5 illustrates a set of stages

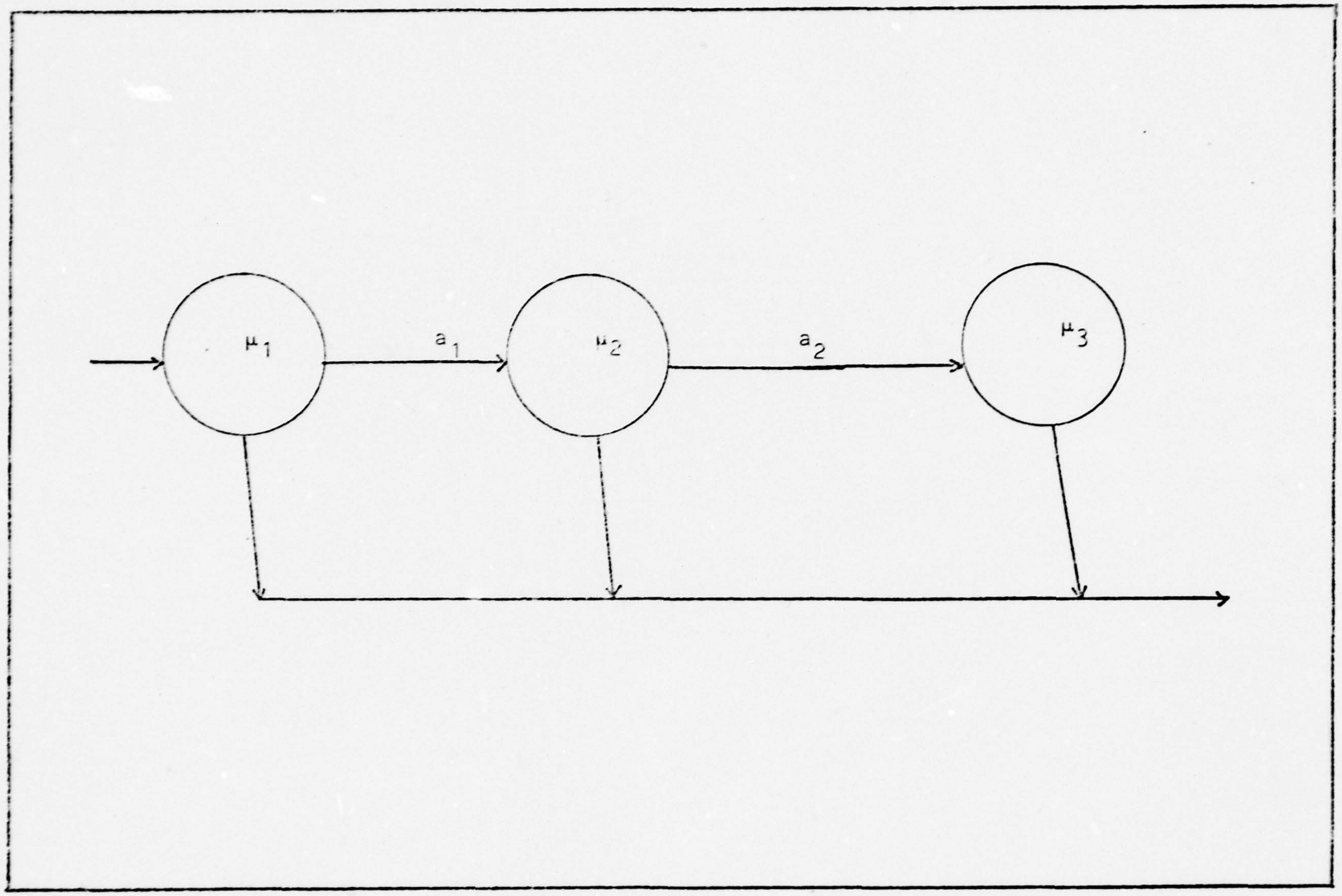


Figure 5. Distribution Represented by Exponential Stages

which may be used to represent the service time distribution of a service center. The probability that a customer receiving service at the ith center is in the lth stage of service is given by (Ref 11):

$$A_{irl} = \prod_{j=1}^{l} a_{irj} \qquad (1)$$

Using the concept of classes of customers and the method of stages, Baskett, et. al. present the equilibrium state solution for a queueing network with four allowable types of service center. These four types of service centers are (Ref 11):

Type 1: Single server, First-Come-First-Serve (FCFS) queueing discipline, all classes must have the same exponential service time distribution.

Type 2: Single server, Processor Sharing (PS) queueing discipline, different classes of customers may have different service time distributions. All service time distributions must have rational Laplace Transform.

Type 3: Infinite servers (IS), no customer waits for service since there are at least as many servers as customers. Different classes of customers may have unique service time distributions so long as the distributions have rational Laplace Transforms.

Type 4: Single server, Preemptive-Resume Last-Come-First-Serve (LCFS) queueing discipline, each class may have a distinct service time distribution with rational Laplace Transform.

Service centers with multiple servers and queue dependent service rates will be considered after the development of the solution for the network of single servers centers.

## III.2 Network State Representation

The state of a model may be described at various levels of abstraction. For example the state may be specified by the total number of customers in the system, the number of customers at each resource, or a detailed description of the ordering and characteristics of the customers at each resource.

Baskett et. al. represent the state of the model by a vector $(x_1, x_2, \ldots, x_M)$ where $x_i$ represents the condition of the ith service center. The method of describing a center's condition depends on the type of center.

If the service center is FCFS then

$$x_i = (x_{i1}, x_{i2}, \ldots, x_{in_i})$$

where $n_i$ is the number of customers at center i and $x_{ij}$ is the class of customer which is in the jth position of the FCFS order. The single server center is considered at this point. Therefore, only the first customer in the queue is served while the remainder are waiting.

If the service center is either PS or IS, the center's condition is described by:

$$x_i=(v_{i1},v_{i1},\dots,v_{iR})$$

where $v_{ir}$ is a vector $(m_{1r},m_{2r},\dots,m_{u_{ir}r})$. The lth component of $v_{ir}$, is the number of class r customers in center i and in the lth stage of service. Each class of customer may have its service time distribution represented by a different number of stages. Therefore, $u_{ir}$ is the number of stages for class r customers at the ith center. All customers in the PS center continually receive some amount of service from the single server. The percentage of full server capacity which each customer receives is dependent upon the total number of customers at the center. Customers at an IS service center receive full server capacity since there are at least as many servers as customers.

If the service center is governed by Preemptive-Resume LCFS discipline, then the center's condition is described by:

$$x_i=((r_1,m_1),(r_2,m_2),\dots,(r_{n_i},m_{n_i}))$$

where $n_i$ is the number of customers at the ith center and $(r_j,m_j)$ describes the jth customer in LCFS order; $r_j$ is the class of the customer and $m_j$ is the stage of service the customer occupies.

The state description for a closed queueing network with multiple classes is much more complex than that used for the single class model described in Appendix A. For the system with multiple classes, the state description must consider all partitions of the customers among the service centers and the ordering and characteristics of the customers at each center. The concept of an equilibrium condition however, is very similar. The equilibrium state probabilities must satisfy the following (Ref 11):

$$\forall \text{ states, } S_i \sum_{\substack{\text{all states}\\ S_j}} P(S_j)[\text{rate of flow from } S_j \text{ to } S_i] = P(S_i)[\text{rate of flow out of } S_i]$$

Chandy (Ref 16) calls these equations the global balance equations. He also defines a set of local balance equations which require that each term on the right-hand side be equal to a particular subset of terms on the left-hand side. Thus, "a local balance equation equates the rate of flow into a state by a customer entering a stage of service to the flow out of that stage due to a customer leaving that stage of service" (Ref 11;252). The examples presented by Baskett, et. al. specify a local balance equation for each class of customer and each service center where each center has only one stage of service.

The solution for the multiple class model yields a set of simultaneous equations similar to those found for the single class model. These are presented by Wong as (Ref 17):

$$\sum_{j=1}^{M} \sum_{s=1}^{R} e_{js} p_{js;ir} = e_{ir} \qquad \begin{matrix} i=1,2\ldots M \\ r=1,2\ldots R \end{matrix} \qquad (2)$$

where $e_{ir}$ is equivalent to $\mu_{ir}X_{ir}$ in the notation used in Appendix A.

The steady state solution for the detailed states is presented as (Ref 11;253):

$$P(x_1, x_2, \ldots, x_M) = Cf_1(x_1)f_2(x_2)\ldots f_M(x_M) \qquad (3)$$

where C is the normalizing constant which causes the steady state probabilities to sum to one over all feasible states. Each function, $f_i$, is dependent upon the type of queueing discipline at service center i. For a FCFS service center,

$$f_i(x_i) = \left[\frac{1}{\mu_i}\right]^{n_i} \prod_{j=1}^{n_i} [e_{ix_{ij}}] \qquad (4)$$

For a PS service center,

$$f_i(x_i) = n_i! \prod_{r=1}^{R} \prod_{l=1}^{u_{ir}} \left[\frac{e_{ir}A_{irl}}{\mu_{irl}}\right]^{m_{irl}} \frac{1}{m_{ikl}!} \qquad (5)$$

For an IS service center,

$$f_i(x_i) = \prod_{r=1}^{R} \prod_{l=1}^{u_{ir}} \left[\frac{e_{ir}A_{irl}}{\mu_{irl}}\right]^{m_{irl}} \frac{1}{m_{irl}!} \qquad (6)$$

For a Preemptive-Resume LCFS service center,

$$f_i(x_i) = \prod_{j=1}^{n_i} e_{ir_j} A_{ir_j m_j} \frac{1}{\mu_{ir_j m_j}} . \qquad (7)$$

The specification of all feasible states using the detailed state description is extremely time consuming since all combinations of customers must be considered as well as the permutations of customers within a particular service center. In order to alleviate this problem, Baskett, et. al. (Ref 11) have developed an aggregate state description. The aggregate state is specified by $S_i = (y_1, y_2, \ldots, y_M)$; where $y_i = (n_{i1}, n_{i2}, \ldots, n_{iR})$ and $n_{ir}$ is the number of class r customers at service center i. The workload parameter for the multiple class model is $N = (n_1, n_2, \ldots, n_R)$. The steady state probabilities are described by the product form solution.

$$P(S=y_1,y_2,\ldots,y_M) = Cg_1(y_1)g_2(y_2)\ldots g_M(y_M) \qquad (8)$$

where, if the service center is FCFS then,

$$g_i(y_i) = n_i! \prod_{r=1}^{R} \frac{1}{n_{ir}!} \left[e_{ir}\right]^{n_{ir}} (1/\mu_i)^{n_i} \qquad (9)$$

if the service center is PS or Preemptive-Resume LCFS then,

$$g_i(y_i) = n_i! \prod_{r=1}^{R} \frac{1}{n_{ir}!} \left[\frac{e_{ir}}{\mu_{ir}}\right]^{n_{ir}} \qquad (10)$$

if the service center is IS then,

$$g_i(y_i) = \prod_{r=1}^{R} \frac{1}{n_{ir}!} \left[\frac{e_{ir}}{\mu_{ir}}\right]^{n_{ir}} \tag{11}$$

The $g_i(Y_i)$ are determined by summing the $f_i$ over all states which have $n_i$ customers at center i. C in Eq (8) is the normalization constant which insures that the sum of the probabilities over all states will equal one. The normalization constant is dependent upon the nature of the workload, therefore:

$$C^{-1}(N) = \sum_{\substack{\text{all feasible} \\ \text{states} \\ y_1 + y_2 + \ldots + y_M = N}} g_1(y_1) g_2(y_2) \ldots g_M(y_M) \tag{12}$$

One form of queue dependent service distribution will be of special interest. Let $H_{ir}(S_i)$ be the rate at which one server at the ith center is serving class r customers when $S_i = (n_{i1}, n_{i2}, \ldots, n_{iR})$. When $H_{ir}(S_i) = f_i(n_i)$, where $f_i(n_i)$ is an arbitrary positive function of the total number of customers at the ith center, then the $g_i(S_i)$ in Eq (8) are replaced by:

$$g_i'(S_i) = g_i(S_i) / \prod_{j=1}^{n_i} f_i(j) \tag{13}$$

This form may be used with service center types 1, 2, or 4 (Ref 17;31). Other forms of the queue dependent service rates are presented by Baskett, et. al. (Ref 11).

## III.3 Consequences of Multiple Classes

An important assumption of the solution for the single class model, Appendix A, is that the traffic intensity for a given workload remained constant in the steady state condition. This assumption permitted the introduction of a new variable, $Q(n_1,n_2,\ldots n_M)$ in Eq (32). The traffic intensity is defined as $X_i = \lambda_i/\mu_i$. The mean service time for a server at the ith service center is given by $1/\mu_i$, and the mean arrival rate of customers to that center is $\lambda_i$. For a computer system, $\frac{1}{\mu_i}$ is the mean time the server (resource) at center i takes to satisfy the requirements of a customer if the full capacity of the server is devoted to that customer. It is assumed that this time does not change with the workload or the number of customers in the queue.

When more than one class is considered, the traffic intensity for each class becomes, $X_{ir} = \lambda_{ir}/\mu_{ir}$. The mean service time and mean arrival rate are now specified for each class at the ith service center. The service time is still dependent upon the requirements of the customer and the capacity of the server. However, the arrival rate may now show a dependence upon the number of customers from other classes present at the queue and on the queueing discipline. As noted earlier, in the steady state condition, the number of customers of a particular class arriving at the ith service center must equal the number in that class departing the center. When processor sharing is used, the number of class r departures is given by:

$$P(S_i)\mu_{ir}\frac{n_{ir}}{n_i}$$

where $n_{ir}/n_i$ is the fraction of the server capacity devoted to class r customers. An identical expression may be used for a FCFS service center. In this case, $n_{ir}/n_i$ represents the probability that a class r customer is the single customer receiving service at the ith center. For the general case, the probability that class r customers are receiving service is given by:

$$\frac{n_{ir}/\ ir}{\sum_{s=1}^{R} n_{is}/\ is}$$

which specifies the fraction of total service requirement devoted to class r customers. For the single class of customer model or when customers have identical service time requirements, this fraction reduces to the earlier form used for the PS service center. The solutions to the balance equations are valid only if the customers at a FCFS service center have identical service time requirements. When the service time requirements at a FCFS center are not identical, the local balance equations are inconsistent (Ref 11;253).

Although a closed form solution is not possible using the technique of local balance, some form of approximation may be possible. Iterative procedures have been used to give approximate results for a fixed workload. Baskett, et. al. (Ref 11) used the predicted arrival rates to compute the mean service time and transition probabilities for an "average" customer. Chen (Ref 18) used an iterative approach to predict performance for a model with queue dependent transition probabilities.

When customers at a FCFS service center do not have identical service time distributions, then some correction to the mean service time of each class is required. Consider two classes of customers; one with a large mean service time, the other with a small mean service time. The time in the queue for the large job will be less than it would be if all the service times were large. This arises from the smaller wait time required while a small job is serviced. The opposite is true for small jobs. They will tend to wait longer in the queue than would be required if all the jobs were small. Therefore the mean service time for large jobs needs to be decreased and the mean service time for small jobs needs to be increased if accurate predictions are to be made. The amount of change will depend upon: the amount of server capacity devoted to each class and the magnitude of the differences in service requirements.

## III.4 Service Time Adjustment

The applicability of the product form solution depends in large part upon how well the restrictions on the queueing types are adhered to. In particular, the restriction requiring equal service time distributions for each class of customer at a FCFS service center will generally not be met at a service center with a Round-Robin scheduling discipline. Even though customers may be allocated equal time quantum for processing at the resource, the amount of service required on the last visit to the server will usually vary with different customers, or even with customers in the same class. Consider a workload consisting of two classes of customers; the first class of customers have a mean service requirement of 7 units while the second class has a requirement for 70 units of service. If the scheduling algorithm assigns a quantum of 7 units to each class of customer during each visit, then the service distribution requirements for a FCFS service center are satisfied. In this instance, the probability that a class r customer is receiving service at the ith service center is $n_{ir}/n_i$. However, if the scheduling algorithm assigns a time quantum of 70 units to both classes, then this simple expression for the probability of receiving service is no longer applicable. In general, the probability of a class r customer being the one receiving service at a FCFS single server center is expressed as the ratio of total class r service required at that resource divided by the total service required by all classes at the resource.

The involved scheduling procedures implemented on the DECsystem-10 introduce additional complications which hinder the direct application of a closed queueing network model. This is especially true when the system is in the dual processor configuration. The customers in PQ1, with their small time quantum allocation, receive preferential assignment to CPU0, the primary processor. Customers in PQ2 are scanned first for allocation of CPU1, the secondary processor. In addition, the count for in-core-protect time may cause a customer to be requeued even though its allocated processing quantum has not expired. For the synthetic jobs used for this validation study, the in-core-protect time expiration caused requeueing only during the first time quantum in PQ2.

The service requirements of a customer in PQ2 can be viewed as consisting of the portion which uses the full quantum allocations and the remainder portion for the last pass through the round robin cycle. Each part of the total PQ2 service requirement can be considered as an individual customer to the FCFS service center. The interaction of the partial service requirements for each class of customer causes some portion to remain in the queue longer than expected if true Processor Sharing where used. Similarly, other portions may complete service faster than expected for a PS service center.

The difference in time spent in the queue associated with the specific resource will depend upon the probability of the occurance of a particular partial service requirement. For the example discussed above, the service requirements may be divided as follows: 7 units for the first class and 30, 30, and 10 units of service for the second class. This division assumes a time quantum of 30 units for both classes. The additional time required for a customer from the first class will depend upon the amount of time that the resource devotes to class 2 customers in either the 30 unit portion or the 10 unit portion. Without a rigorous mathematical solution to the problem, a logical approach is to assume that the total required service time adjustment is equally divided ahead of and behind the service portion in question. Therefore

$$S'_{irm} = S_{irm} + \sum_{s=1}^{R} \sum_{k=1}^{u_s} U_{is} P_{isk} (S_{irk} - S_{irm})/2 \qquad (14)$$

where

$S_{irm}$ = Service required at the ith center by class r customers receiving the mth portion of service

$U_{is}$ = Utilization of ith service center by class s customers

$P_{isk}$ = Probability that class s customer is in kth portion of service

$u_s$ = Number of divisions which comprise the service time requirement for class s customers.

Large correction factors result when a customer has a small remainder portion while there are many customers which use full quantum allocations. This point is illustrated by a workload consisting of a single class of customers with mean service requirement of 31 units. If the service time quantum is set at 30 units, then the customers will have to process through the queue a second time to receive the final unit of service. The total time at the service center may be much longer than it would have been if the total service requirement were 29 units. The consequences of relatively small changes in mean service requirements on performance measurements are presented in Chapter V.

## III.5 Summary

Baskett, et. al. (Ref 11), developed a solution for the Closed Queueing Network Model with multiple classes of customers. Although this solution permits more flexibility for representing an actual computer system, there are assumptions required for the solution which are not totally satisfied in real world systems. This chapter has presented a discussion of the solution developed by Baskett, et. al. and has indicated the physical implications of the assumptions. In addition, an extensive discussion of the restrictions on FCFS service centers is presented. In this final discussion, the limitations of the solution, as they apply to the DECsystem-10, have been emphasized.

# Chapter IV

# Computer Performance Measures

The solution of the equilibrium state probabilities presented in Chapter III permits the calculation of system performance measures such as marginal queue length distributions for each queue, resource utilization, and mean response time. This chapter presents computationally efficient algorithms for computing the mean arrival rate, $_{ir}$, and the resource utilization, $U_{ir}$. These algorithms are not applicable to all the service centers discussed in Chapter III, but they may be used with many models of interest so that computation requirements may be reduced.

## IV.1 Normalization Constant Calculation

In order to obtain the equilibrium state probabilities, the $g_i(S_i)$ must be calculated for all the service centers and all states of the center. In addition, the normalization constant must be determined by summing the product form solution over all feasible states. If C(N) is computed in a straight-forward manner, the amount of computation increases proportional to the total number of states in the model. If the network is closed and the customers do not change class, and every customer visits every service center, then the number of states is given by (Ref 17;48):

$$\prod_{r=1}^{R} \binom{M+N_r-1}{M-1}$$

For models of interest, the straight-forward computation of C(N) may become impractical. Wong (Ref 17) and McKenzie (Ref 1) present a computationally more efficient method for determining C(N). Two restrictions are placed on the use of this method (Ref 17;47):

1) Customers do not change class

2) Every class of customer visits every service center

Adjustments may be made in the network's state description to overcome the first restriction, while relatively minor changes in the computational method will permit customer classes which do not visit every service center. The program developed by Mckenzie (Ref 1) requires that both restrictions be met. For the purpose of this thesis, these restrictions were not unreasonable. More detailed models, those modeling a greater variety of system resources, may require that class changes be permitted or that certain classes do not visit all service centers. These models will require that the program be modified to include the alternate procedures suggested by Wong (Ref 17;60).

## IV.2 Queue Length Calculation

After the equilibrium state probabilities and normalization constant have been calculated, the marginal queue length distributions and expected number of customers can be computed for each resource and each class of customer. The probability that there are $n_i$ customers at the ith service center is given by (Ref 17;54):

$$P_i(n_i) = \sum_{\substack{\text{all } S \\ \text{such that} \\ n_{i1}+n_{i2}+\ldots+n_{iR}=n_i}} P_i(S_i) \tag{15}$$

The probability that there are $n_{ir}$ class r customers at the ith service center is given by:

$$P_i(n_{ir}) = \sum_{\substack{\text{all states} \\ \text{such that there} \\ \text{are } n_{ir} \text{ class r} \\ \text{customers at service} \\ \text{center i}}} P_i(S_i) \tag{16}$$

The expected number of total customers, $n_i$, and expected number of class r customers, $n_{ir}$, at the ith service center are easily computed from Eqs (15) and (16).

## IV.3 Resource Utilization Calculation

For a single server service center, resource utilization, $U_{ir}$, is the fraction of a unit time interval that the server at ith center is busy serving class r customers. When the service center is type 1, 2, or 4, $U_{ir}$ is computed by:

$$U_{ir} = \sum_{\text{all } S_i} P_i(S_i) \frac{n_{ir}}{n_i} \qquad (17)$$

where $n_{ir}/n_i = 0$ when $n_{ir}$ and $n_i$ are both zero. This ratio is the fraction of server capacity received by class r customers when the ith center has a PS discipline. For FCFS or Preemptive-Resume LCFS centers, the ratio is the probability that a class r customer is getting full capacity of the ith service center. This assumes that the service time distributions meet the restrictions described in Chapter III.

If the service center is single server and has a constant service rate, then a simplier expression for resource utilization is possible. Resource utilization may be expressed as (Ref 17;55):

$$U_{ir} = \lambda_{ir}/\mu_{ir} \qquad (18)$$

Wong (Ref 17) expands on this expression after he proves that the arrival rate is given by:

$$\lambda_{ir} = \frac{C(N)}{C(N^{-r})} e_{ir} \qquad (19)$$

where $N^{-r} = (N_1, N_2, .., N_r - 1, .., N_M)$, which permits the utilization to be expressed by:

$$U_{ir} = \frac{C(N)}{C(N^{-r})}\left[\frac{e_{ir}}{\mu_{ir}}\right] \tag{20}$$

When service center i contains $a_i$ identical resources, the utilization, $U_{ir}$, is the average measure of the utilization of the individual resources. Therefore:

$$U_{ir} = \frac{\lambda_{ir}}{a_i \mu_{ir}} \tag{21}$$

If the service center has a PS discipline, then the following expression holds, even for multiple classes of customers:

$$U_{ir} = \frac{C(N)}{C(N^{-r})}\left[\frac{e_{ir}}{a_i \mu_{ir}}\right] \tag{22}$$

For service centers of type 1 or 4, this expression does not apply. The utilization, $U_{ir}$, must be computed by Eq (17). The straight-forward approach must also be used when the servers have state dependent service rates. The processors on the DECsystem-10 at the Air Force Avionics Laboratory are not identical. The operating system overhead on the primary processor is consistently higher than that of the secondary processor. In addition, the secondary must stop processing a customer which initiates an I/O request since only the primary has the normal I/O capability. Because of these features, the model for the primary/secondary configuration should be state dependent in order to accurately reflect the physical system.

The user terminals in a time-sharing computer system are represented by an Infinite Server (IS) service center. This type of service center has an expression for resource utilization similar to the expressions for the other service center types. In an IS service center, there are at least $N_r$ servers dedicated to the $N_r$ class r customers in the system. If the IS service center were represented as having $N_r$ identical servers then the utilization would be $\lambda_{ir}/N_r\mu_{ir}$. However it is notationally more convenient to express the utilization as (Ref 17;58):

$$U_{ir} = \frac{\lambda_{ir}}{\mu_{ir}} \tag{23}$$

where $U_{ir}$ will be the mean number of class r customers at the ith center when it is an IS service center. Then the utilization can be expressed as:

$$U_{ir} = \frac{C(N)}{C(N^{-r})}\left[\frac{e_{ir}}{\mu_{ir}}\right] \tag{24}$$

These expression for $\lambda_{ir}$ and $U_{ir}$ allow for more efficient computation of the mean arrival rate and resource utilization. The values of $C(N^{-r})$ are derived in the iterative process used by McKenzie (Ref 1) to calculate the normalization constant. When the service center is of the type discussed above, the values of $C(N^{-r})$ can be retained and the less efficient computations required by Eq (17) can be avoided.

## IV.4 Response Time Calculation

When $\lambda_{ir}$ has been calculated, the response time for class r customers can be determined. Response time is the amount of time a customer spends in the network outside of the IS service center representing the terminals. This is shown in Figure 6. The mean number of class r customers "thinking" at the terminals is given by $n_{ir}$. The number of class r customers in the remainder of the network is $N_r - n_{ir}$. From Little's formula (Ref 19), the mean response time is given by:

$$T_{ir} = \frac{N_r - n_{ir}}{\lambda_{ir}} \tag{25}$$

## IV.5 Summary

The number of possible states for a Closed Queueing Network Model increases considerably with the addition of service centers or the specification of multiple classes. The direct computation of the normalization constant of the product form solution and resource utilization requires that specific values be summed over all possible states. In order to reduce the computational requirements for solution of these models, alternate methods have been presented to determine the normalization constant, mean arrival rate and

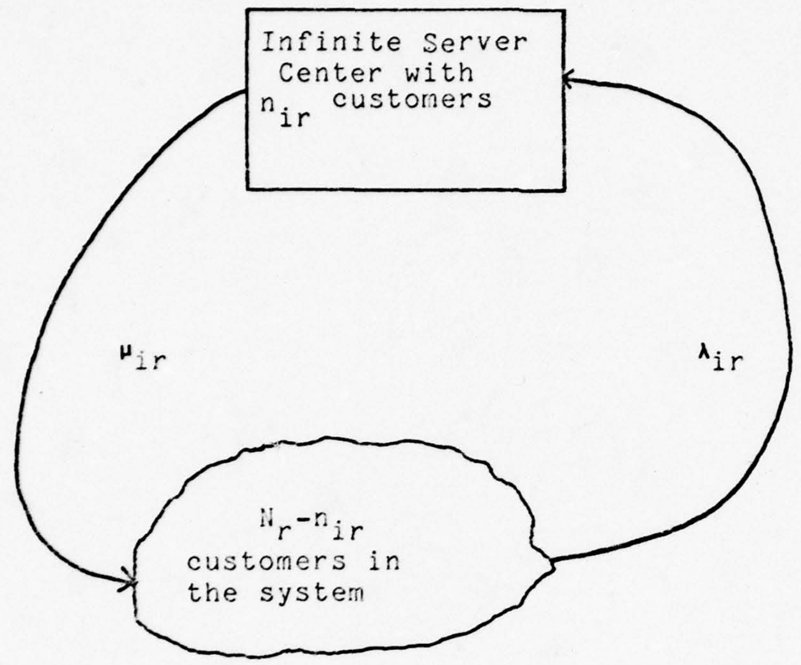

Infinite Server Center with $n_{ir}$ customers
$\mu_{ir}$
$\lambda_{ir}$
$N_r - n_{ir}$ customers in the system

Figure 6. Response Time Calculation

resource utilization. These methods do require that additional assumptions be made concerning the model. However, they are applicable to most models of interest.

# Chapter V

# Model Validation

The concept of model validation is not easily defined. However, "the validation ought to establish some degree of confidence that the model produces results which sufficiently mirror the significant attributes of the system being modeled (Ref 21;139)." The time limitations of this work prohibit the use of detailed hypothesis testing techniques which should be used to make a final model selection. Since this report is primarily concerned with performance for time-sharing users, value judgments based on response time errors will be the basis for evaluating the quality of a model.

It is not possible to validate all combinations of system characteristics with absolute certainty. In any case, many of the resource configuration and workload combinations would be of little value to understanding the real system. The validation effort should be directed towards common system configurations and reasonable workload characterizations. The model validation effort investigates the application of several Closed Queueing Network Models to the DECsystem-10 dual processor system. In addition, the single processor configuration is examined as well as the application of a model to represent the system when swapping occurs.

The validation of the Closed Queueing Network Model is performed with the use of a synthetic workload. The workload is developed using the SCRIPT program to run duplicate copies of selected synthetic jobs. Each job cycles through a predetermined sequence of subroutine calls. The cycle starts with the job in a sleep state which is used to model teletype think time. After the sleep state, a CPU subroutine is called a specified number of times. Finally, the job makes a specified number of calls to a disk I/O subroutine which writes a single block of data onto a designated RP03 disk file. When the disk I/O is completed, the job returns to the sleep state which represents user think time. Numerous combinations of processing and I/O requirements are possible with synthetic jobs of this form. One restriction of the synthetic jobs which is not required of real jobs is that processing and disk I/O are sequential. Although processing and I/O may overlap when different jobs are involved, processing and I/O within a single job do not overlap. This restriction on the synthetic workload coincides with the queueing model restriction prohibiting a customer from using two resources at one time.

The overhead introduced by the Operating System is represented by increasing the processing service requirements by the amount of service time which would be lost to overhead. Because the Operating System overhead is not equally divided between the two processors, they will not be considered as identical servers. The use of state dependent service rates does permit realistic modeling of this aspect of the processors. Customers in PQ2 are considered first for assignment to the secondary processor. Therefore the overhead of CPU1, approximately 5%, is used to determine the increase in service time requirements of each customer in PQ2. The overhead of the primary processor is much higher, approximately 12%. Because of the larger overhead on CPU0, the service rate is multiplied by 1.93 when two or more customers are present in the queue. The service time requirement of customers in PQ1 is increased by an amount determined by CPU0 overhead.

This validation study considers two basic questions raised by Giammo (Ref 21):

> 1) Do the service time assumptions introduce a basic distortion?
>
> 2) Can the logic of job processing be sufficiently modeled as a network of queues?

The validation effort considers numerous workload combinations and the basic processor configurations available on the DECsystem-10. Although disk I/O is included in the workload requirements, the main emphasis of this work is directed towards accurately modeling the processing capabilities of the system. The involved scheduling and queueing procedures associated with the processors justify detailed analysis of this area. A total of 21 experiments were conducted to validate the models examined in this report.

Each validation experiment considers a specific model configuration and combinations of two or three synthetic workloads. The workloads consist of various combinations of customers from one or two classes. Each customer class has predetermined processing and disk I/O requirements. The individual experiments will analyze the effects of different numbers of customers from each class and changes in the processing requirements of the classes. For the jobs executed in this validation work, the DECsystem-10 run queue is considered in two portions, PQ1 and PQ2. Since jobs in each queue are handled differently by the scheduler, a natural question concerns how processor queueing should be represented. The models evaluated in this report use two seperate service centers in one case and in the other, a single service center to represent the queues which form at the processor resource.

## V.1 Two Node Processor Model

Experiments 1 through 6 investigated the effect of different workloads on a specific model of the DECsystem-10 in the dual processor configuration. Each experiment considers combinations of the synthetic jobs with predetermined processing and I/O requirements. The model configuration used for this series of experiments is the multiple server model used by Mckenzie (Ref 1). This model, shown in Figure 7, represents the teletype by an

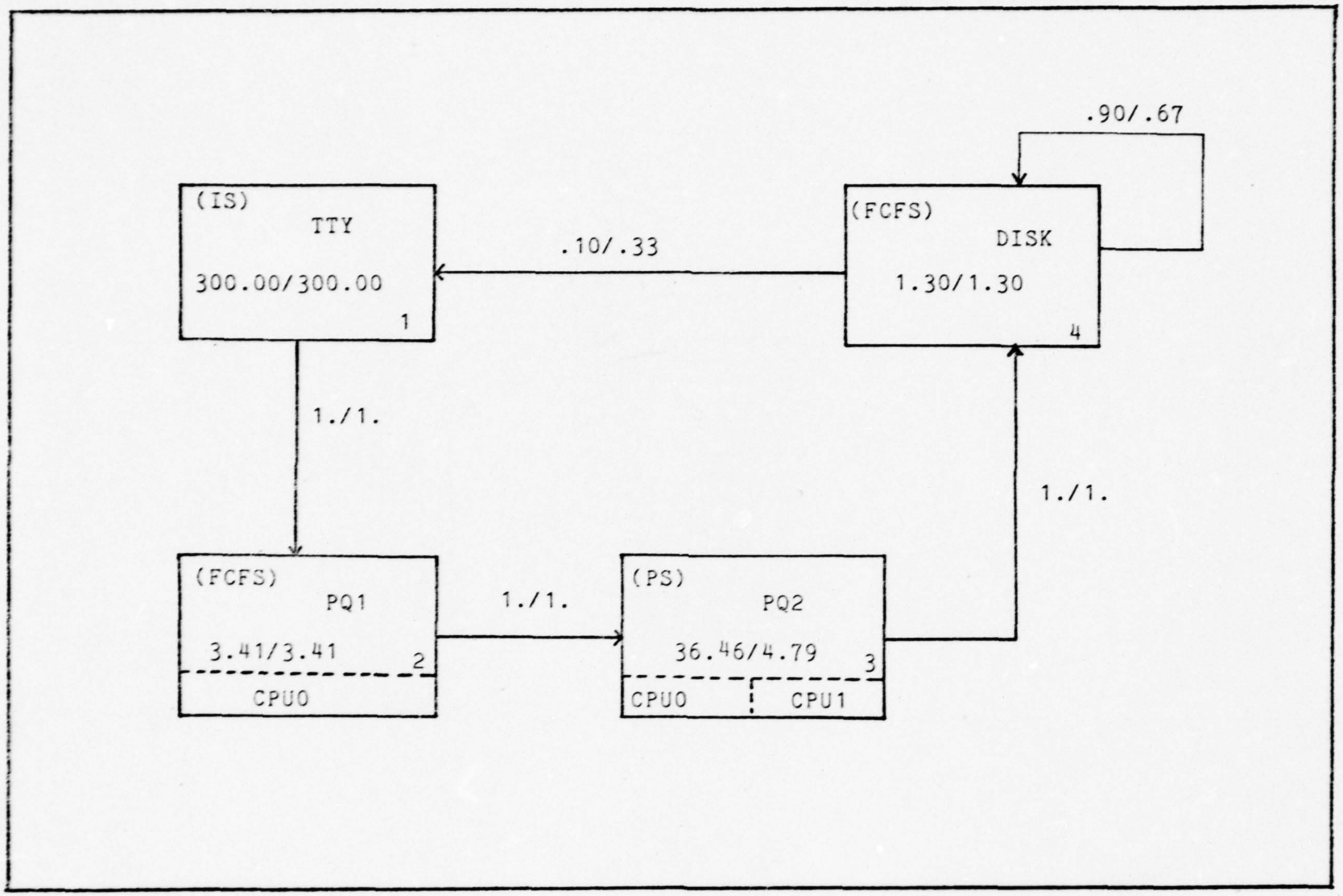


Figure 7. Two Node Processor Model

Infinite Server service center. The processors are represented by a serial combination of two service centers. A single server FCFS center services customers in PQ1 as they depart the teletype node. Customers in PQ2 are serviced by a multiple server Processor Sharing center. Disk I/O is represented by a single server FCFS center with feedback to represent multiple disk calls. The service times and transition probabilities in Figure 7 are those used for experiment 6.

### V.1.1 Experiment 1

Experiment 1 investigates the application of the model shown in Figure 7 for a workload which consists of a single class of job with the following characteristics:

| | | |
|---|---|---|
| Think Time | = | 300.0 jiffies |
| Processor Service | = | 7.6 jiffies |
| Disk I/O Calls | = | 3 |

The results of this experiment are shown in Table I, where the workload consisted of 20 jobs for part 1 and 10 jobs for part 2. The workload closely matches the service time distribution requirement placed on a FCFS service center. Although the model uses a Processor Sharing service center to represent the processing in PQ2, the real system's round-robin scheduling more closely resembles a FCFS center with feedback, as discussed in Chapter III. This concept of the actual system will permit a better understanding of the results of this validation study. Although the equal service

Table I

Experiment 1 Results

| | Mean Response Time | | Mean Number in Queue | | | Utilization | |
|---|---|---|---|---|---|---|---|
| | Class 1 | Class 2 | Disk I/O | Run | TTY I/O | CPU0 | CPU1 |
| Experiment 1.1 | | | | | | | |
| Measured | 14.72 | N/A | 0.40 | 1.50 | 18.10 | 40.40% | 30.40% |
| Predicted | 14.36 | | 0.33 | 0.59 | 19.08 | 26.72% | 26.72% |
| Error | 2.45% | | 17.50% | 60.67% | 5.41% | 33.91% | 12.17% |
| Experiment 1.2 | | | | | | | |
| Measured | 13.22 | N/A | 0.15 | 1.40 | 8.45 | 33.20% | 10.40% |
| Predicted | 13.04 | | 0.14 | 0.28 | 9.58 | 14.34% | 14.34% |
| Error | 1.36% | | 6.67% | 80.00% | 13.37% | 57.93% | 37.50% |

times of the jobs would seem to exactly satisfy the requirements for a FCFS service center, the sequencing of jobs through PQ1 and PQ2 may cause the jobs to have differing service requirements depending on which cycle of the Round-Robin sequence they are in. In this example each job will receive 3 jiffies of service while in PQ1 and the remaining 4.6 jiffies while in PQ2; service time compensation for overhead will be ignored for this discussion. The minimum PQ2 time quantum is larger than the service requirement remaining after the job leaves PQ1. Therefore, each job will receive two relatively equal bursts of service from the processors, one of 3 jiffies and the other of 4.6 jiffies. This factor contributes to the extremely low response time error shown for both workloads.

### V.1.2 Experiment 2

The workload for experiment 2 consists of a single class of customers with the following characteristics:

| | | |
|---|---|---|
| Think time | = | 300.0 jiffies |
| Processor Service | = | 30.6 jiffies |
| Disk I/O Calls | = | 10 |

The results of this experiment are shown in Table II for 20 jobs and 10 jobs in part 1 and part 2 respectively. The response time error has increased for both workload levels. The processing requirement of the jobs in this experiment produce processing cycles of differing length, even though the total service times of the jobs are identical. The PQ1

Table II

Experiment 2 Results

| | Mean Response Time | | Mean Number in Queue | | | Utilization | |
|---|---|---|---|---|---|---|---|
| | Class 1 | Class 2 | Disk I/O | Run | TTY I/O | CPU0 | CPU1 |
| Experiment 2.1 | | | | | | | |
| Measured | N/A | 105.26 | 0.85 | 4.60 | 14.55 | 97.10% | 98.80% |
| Predicted | | 91.61 | 1.67 | 3.01 | 15.32 | 86.93% | 86.93% |
| Error | | 12.97% | 96.47% | 34.57% | 5.29% | 10.50% | 12.04% |
| Experiment 2.2 | | | | | | | |
| Measured | N/A | 47.54 | 0.30 | 1.90 | 7.80 | 51.20% | 61.80% |
| Predicted | | 56.20 | 0.53 | 1.05 | 8.42 | 59.42% | 59.42% |
| Error | | 18.22% | 76.67% | 44.74% | 7.95% | 16.02% | 3.88% |

time slice is 3 jiffies, while the remainder received in PQ2 is 27.6 jiffies. The assumption of the model is that all jobs either receive equal service of the resource, for Processor Sharing, or that the probability of receiving service is equal for all jobs, for a FCFS resource. The differing quantum allocations prevent this assumption from being met. However, the priority assignment of jobs in PQ1 to CPU0, and jobs in PQ2 to CPU1 does compensate somewhat by providing for jobs in PQ1 a more equal probability of receiving service. McKenzie (Ref 1) discusses one procedure to help alleviate this problem. In effect, the time that jobs in PQ2 wait for jobs in PQ1 is added to the total service time. However, this procedure does not address all aspects of the problem. The service time needs to be decreased by some amount because jobs in PQ2 on the real world system must wait for a job with a time slice of only 27.6 rather than the 30.6 assumed by the model. Offsetting this adjustment is the added time that jobs in PQ1, with a 3 jiffy time slice, must wait for jobs from PQ2 which may have been assigned to both processors. The effect of each of these factors is difficult to determine, but it would seem to depend upon how busy the processors are with each type of job. For this particular experiment, the model predictions are less than measured when the processors are very busy; while the predictions are higher than measured when there is low processor utilization.

A final consideration which applies to all experiments which display relatively low processor utilization is that the deterministic nature of the synthetic jobs does not guarantee that a true steady state condition is obtained. In fact, observations of the processor control display during these experiments indicated that jobs tended to process in "bunches" when the processor utilization was approximately 85% or lower. This observation was further substantiated by the METER reports which indicated that the response time of the synthetic jobs tended to be in one of two clusters under this workload condition. The measured response time for each cycle of the synthetic job in experiment 2.2 fell in a groups centered at approximately 135 jiffies and 75 jiffies. If the conditions of the model were completely satisfied, the measured response times would be more closely distributed about the overall mean.

### V.1.3 Experiment 3

The workload for experiment 3 also consists of a single class of customer with the following characteristics:

| | | |
|---|---|---|
| Think Time | = | 300.0 jiffies |
| Processor Service | = | 83.6 jiffies |
| Disk I/O Calls | = | 3 |

The results of this experiment for 20 and 10 jobs are shown in Table III part 1 and part 2 respectively. The response time errors are similar to those shown for experiment 2. For this experiment, the longer service time required at the processor permits each job to receive more time quanta while

Table III

Experiment 3 Results

| | Mean Response Time | | Mean Number in Queue | | | Utilization | |
|---|---|---|---|---|---|---|---|
| | Class 1 | Class 2 | Disk I/O | Run | TTY I/O | CPU0 | CPU1 |
| Experiment 3.1 | | | | | | | |
| Measured | 624.41 | N/A | 0.10 | 14.40 | 5.50 | 100.00% | 100.00% |
| Predicted | 579.25 | | 0.10 | 13.08 | 6.82 | 99.99% | 99.99% |
| Error | 7.23% | | 0.00% | 9.17% | 24.00% | 0.01% | 0.01% |
| Experiment 3.2 | | | | | | | |
| Measured | 154.59 | N/A | 0.05 | 4.10 | 5.85 | 99.70% | 100.00% |
| Predicted | 187.91 | | 0.09 | 3.76 | 6.15 | 95.74% | 95.74% |
| Error | 21.55% | | 80.00% | 8.29% | 5.13% | 4.01% | 4.30% |

in PQ2. The longer service time also causes more jobs to wait in PQ2 and increases the probability that jobs from PQ2 will be using both processors. Because of this, jobs coming out of the sleep state will more likely be required to wait while a job from PQ2 completes service on CPU0, but the overall processor service will be closer to processor sharing.

### V.1.4 Experiment 4

The job mix for experiment 4 consists of two classes of jobs with the following characteristics:

```
Think Time           = 300.0 jiffies
Processor Service
             Class 1 =  38.0 jiffies
             Class 2 =  76.0 jiffies
Disk I/O Calls
             Class 1 =  10
             Class 2 =   3
```

The results of this experiment are shown in Table IV parts 1, 2 and 3 for job mixes of 10 each, 8 each, and 5 each respectively. The use of this more realistic workload provides a clearer picture of the value of the Closed Queueing Network Model. With this workload, the actual computer system has a variety of customers in differing stages of completion in the Round-Robin processing cycle. This provides an even greater departure from the equal service rate requirement for a true FCFS service center. The good error rate for the mix of 5 jobs of each type is believed to be the result of the failure to reach a true steady state condition rather than improved model accuracy.

Table IV

Experiment 4 Results

| | Mean Response Time | | Mean Number in Queue | | | Utilization | |
|---|---|---|---|---|---|---|---|
| | Class 1 | Class 2 | Disk I/O | Run | TTY I/O | CPU0 | CPU1 |
| Experiment 4.1 | | | | | | | |
| Measured | 275.07 | 405.28 | 0.40 | 11.10 | 8.50 | 100.00% | 100.00% |
| Predicted | 195.59 | 355.01 | 0.46 | 8.90 | 10.63 | 99.70% | 99.70% |
| Error | 28.89% | 12.40% | 15.00% | 19.82% | 25.06% | 0.30% | 0.30% |
| Experiment 4.2 | | | | | | | |
| Measured | 173.98 | 255.11 | 0.15 | 7.30 | 8.55 | 100.00% | 100.00% |
| Predicted | 134.17 | 230.69 | 0.41 | 5.54 | 10.05 | 97.91% | 97.91% |
| Error | 22.88% | 9.57% | 173.33% | 24.11% | 17.54% | 2.10% | 2.10% |
| Experiment 4.3 | | | | | | | |
| Measured | 75.80 | 110.32 | 0.10 | 3.20 | 6.70 | 73.40% | 89.60% |
| Predicted | 77.64 | 123.71 | 0.27 | 2.22 | 7.51 | 85.23% | 85.23% |
| Error | 2.43% | 12.14% | 170.00% | 30.63% | 12.09% | 16.08% | 4.91% |

## V.1.5 Experiment 5

Two classes of jobs were also used for experiment 5. These jobs have the following characteristics:

```
Think Time           = 300.0 jiffies
Processor Service
          Class 1 =   30.6 jiffies
          Class 2 =   83.6 jiffies
Disk I/O Calls
          Class 1 =   10
          Class 2 =    3
```

The results of this experiment for job mixes of 10 each, 8 each, and 5 each are shown in Table V parts 1, 2 and 3. A slight improvement in response time error is indicated over similar job mixes in experiment 4. The time quantum allocated for both classes of jobs is more fully utilized than was true for experiment 4. For example, type 2 customers in experiment 4 had a PQ2 remainder cycle of 5 jiffies which competed for the processors with other jobs requiring a full 30 jiffy time slice. For this experiment, the PQ2 service requirement of type 1 jobs, 27.6 jiffies, is satisfied in one visit to the processor. The changes which produced the slight improvement for response time error in experiments 5.1 and 5.2, over similar parts of experiment 4, had a negative effect on the apparent accuracy of the model for the mix of 5 jobs of each class.

## V.1.6 Experiment 6

The final experiment using the model represented in Figure 7, to predict the performance of the DECsystem-10 in

Table V

Experiment 5 Results

| | Mean Response Time | | Mean Number in Queue | | | Utilization | |
|---|---|---|---|---|---|---|---|
| | Class 1 | Class 2 | Disk I/O | Run | TTY I/O | CPU0 | CPU1 |
| Experiment 5.1 | | | | | | | |
| Measured | 202.09 | 442.25 | 0.30 | 10.80 | 8.90 | 100.00% | 100.00% |
| Predicted | 152.03 | 365.62 | 0.51 | 8.24 | 11.14 | 99.74% | 99.74% |
| Error | 24.77% | 17.33% | 70.00% | 23.70% | 25.17% | 0.30% | 0.30% |
| Experiment 5.2 | | | | | | | |
| Measured | 115.50 | 277.92 | 0.15 | 7.00 | 8.85 | 100.00% | 100.00% |
| Predicted | 106.84 | 239.65 | 0.44 | 5.21 | 10.35 | 97.77% | 97.77% |
| Error | 7.50% | 13.77% | 193.33% | 25.57% | 16.95% | 2.30% | 2.30% |
| Experiment 5.3 | | | | | | | |
| Measured | 55.17 | 113.87 | 0.10 | 2.80 | 7.10 | 75.80% | 93.00% |
| Predicted | 64.66 | 132.20 | 0.28 | 2.14 | 7.58 | 84.70% | 84.70% |
| Error | 17.20% | 16.10% | 180.00% | 23.57% | 6.76% | 11.74% | 8.92% |

the dual processor configuration also uses a job mix of two class types. The characteristics of the jobs used in experiment 6 are as follows:

| | | |
|---|---|---|
| Think Time | = | 300.0 jiffies |
| Processor Service | | |
| Class 1 | = | 38.0 jiffies |
| Class 2 | = | 7.6 jiffies |
| Disk I/O Calls | | |
| Class 1 | = | 10 |
| Class 2 | = | 3 |

The results of this experiment are found in Table VI parts 1, 2 and 3 for job mixes of 10 each, 8 each, and 5 each respectively. This experiment considers jobs which place light demands on the system. Only one full PQ2 time quantum is used, that being for type 1 jobs. The remaining portions of service which the jobs receive in the Round-Robin processing cycle are relatively equal. The fact that most of the processing cycles are nearly equal is reflected in the good response time errors.

## V.2 <u>One Node Processor Model</u>

For experiments 7 through 12, the DECsystem-10 dual processors are modeled with a single service center using the Processor Sharing queueing discipline. The total CPU service time requirement, PQ1 time plus PQ2 time, is assumed to be received at this service center, rather than in stages as in experiments 1 through 6. Figure 8 shows this model representation of the DECsystem-10. The service times and transition probabilities in the figure are those used for experiment 12.

Table VI

Experiment 6 Results

| | Mean Response Time | | Mean Number in Queue | | | Utilization | |
|---|---|---|---|---|---|---|---|
| | Class 1 | Class 2 | Disk I/O | Run | TTY I/O | CPU0 | CPU1 |
| Experiment 6.1 | | | | | | | |
| Measured | 86.18 | 27.15 | 0.70 | 3.60 | 15.70 | 69.40% | 84.20% |
| Predicted | 75.95 | 18.01 | 0.82 | 1.78 | 17.40 | 73.92% | 73.92% |
| Error | 11.87% | 33.66% | 17.14% | 50.56% | 10.83% | 6.48% | 12.23% |
| Experiment 6.2 | | | | | | | |
| Measured | 76.55 | 22.69 | 0.15 | 2.60 | 13.25 | 66.00% | 71.50% |
| Predicted | 67.52 | 16.12 | 0.58 | 1.30 | 14.12 | 64.85% | 64.85% |
| Error | 11.80% | 28.96% | 286.67% | 50.00% | 6.57% | 1.82% | 9.37% |
| Experiment 6.3 | | | | | | | |
| Measured | 59.77 | 21.83 | 0.70 | 1.70 | 7.60 | 44.10% | 38.20% |
| Predicted | 58.97 | 14.06 | 0.31 | 0.74 | 8.95 | 46.83% | 46.83% |
| Error | 1.34% | 35.59% | 55.71% | 56.47% | 17.76% | 6.12% | 22.51% |

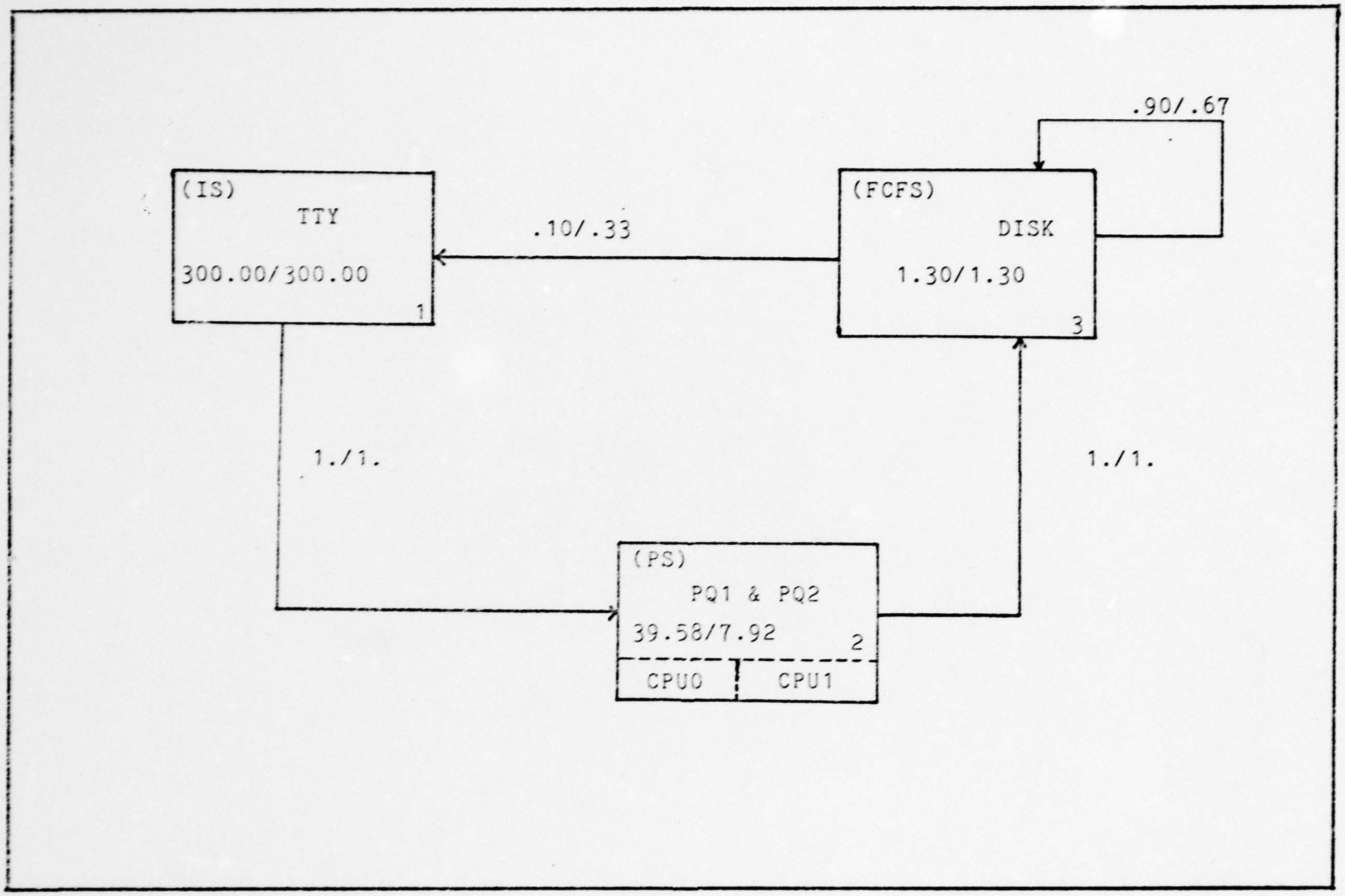


Figure 8. One Node Processor Model

This model does not provide for the priority service which jobs receive while in PQ1. The failure to provide priority service for jobs in PQ1 tends to increase predicted response time because, in the model, these jobs share the total capacity of the processors with jobs from PQ2. This same feature could also cause some decrease in predicted response time since the model allows jobs in PQ2 to be processed by either processor, giving equal capacity to time in PQ1 and PQ2. This feature could provide jobs at the CPU service center with more resource service than would be found in the real world system. Combining the two run queues will have different effects on different jobs processed through the model. The direction and magnitude of error will depend upon such features as number of customers receiving full PQ2 time quantum and the number of customers receiving service while in PQ1. The results of experiments 7 through 12 indicate this dependency upon the type of workload. However, these results also show that this simplied system representation is approximately as accurate as the model which used two nodes to represent the processors.

### V.2.1 Experiment 7

The results of experiment 7 are shown in Table VII. For this experiment, the workload characteristics were the same as used for experiment 1. Although the response time errors have increased for both workloads, the size of the error is still relatively small when compared with results

Table VII

Experiment 7 Results

| | Mean Response Time | | Mean Number in Queue | | | Utilization | |
|---|---|---|---|---|---|---|---|
| | Class 1 | Class 2 | Disk I/O | Run | TTY I/O | CPU0 | CPU1 |
| Experiment 7.1 | | | | | | | |
| Measured | 14.72 | N/A | 0.40 | 1.50 | 18.10 | 40.40% | 30.40% |
| Predicted | 13.66 | | 0.33 | 0.54 | 19.13 | 41.89% | 41.89% |
| Error | 7.20% | | 17.50% | 64.00% | 5.69% | 3.69% | 37.80% |
| Experiment 7.2 | | | | | | | |
| Measured | 13.22 | N/A | 0.15 | 1.40 | 8.45 | 33.20% | 10.40% |
| Predicted | 12.50 | | 0.14 | 0.26 | 9.60 | 22.78% | 22.78% |
| Error | 5.45% | | 6.67% | 81.43% | 13.61% | 31.39% | 119.04% |

of experiments involving more complex workloads. The fact that the predicted response times decreased further below the measured values indicates that, for this experiment, the assumption of the model that all jobs receive an equal share of capacity from the processors is too optimistic. The measured values indicate that there is more conflict for processor service than is predicted by the model.

### V.2.2 Experiment 8

The results for experiment 8 are shown in Table VIII. This experiment used the same workload inputs as experiment 2. The increase in predicted response times indicates that the modeled system has become more congested. This results from the increased service time which causes jobs to remain in the run queue longer and therefore increases the number of jobs in the queue for the processors. Since one of the response time errors decreased while the other increased, it is difficult to make a comparison between the two models. It should be noted again however, that the measured data for light workloads did not appear to be from a true steady state condition. Because of this, all experiments which have low CPU utilizations are in question. These experiments may still be of interest in that they establish some measure of accuracy for this level of workload when a steady state condition is not achieved.

Table VIII

Experiment 8 Results

| | Mean Response Time | | Mean Number in Queue | | | Utilization | |
|---|---|---|---|---|---|---|---|
| | Class 1 | Class 2 | Disk I/O | Run | TTY I/O | CPU0 | CPU1 |
| Experiment 8.1 | | | | | | | |
| Measured | N/A | 105.26 | 0.85 | 4.60 | 14.55 | 97.10% | 98.80% |
| Predicted | | 101.72 | 1.59 | 3.48 | 14.94 | 90.85% | 90.85% |
| Error | | 3.36% | 87.06% | 24.35% | 2.68% | 6.44% | 8.05% |
| Experiment 8.2 | | | | | | | |
| Measured | N/A | 47.54 | 0.30 | 1.90 | 7.80 | 51.20% | 61.80% |
| Predicted | | 57.30 | 0.53 | 1.07 | 8.40 | 63.84% | 63.84% |
| Error | | 20.53% | 76.67% | 43.68% | 7.69% | 24.69% | 3.30% |

## V.2.3 Experiment 9

The results of experiment 9 are shown in Table IX. The workloads used in this experiment were the same as used for experiment 3. The response time errors show changes very similar to those of experiment 8. Since the total CPU service time is larger than used for experiment 8, there are more jobs in the run queue. In the model this results in a lower percentage of server capacity for jobs waiting at the processor service center.

## V.2.4 Experiment 10

The results of experiment 10 are shown in Table X. This experiment used the same workloads as used for experiment 4. For the workloads which cause heavy CPU utilization, the response time errors improved for both classes of customers. For this experiment, the simplified model appears to be at least as accurate as the model used in experiment 4.

## V.2.5 Experiment 11

Table XI shows the results for experiment 11. The workloads for this experiment were the same as used for experiment 5. The response time errors for this experiment show the same trend as noted for experiment 10. The use of one Processor Sharing node is at least as accurate a representation of the DECsystem-10 as the dual node representation used for experiment 5.

Table IX

Experiment 9 Results

| | Mean Response Time | | Mean Number in Queue | | | Utilization | |
|---|---|---|---|---|---|---|---|
| | Class 1 | Class 2 | Disk I/O | Run | TTY I/O | CPU0 | CPU1 |
| Experiment 9.1 | | | | | | | |
| Measured | 624.41 | N/A | 0.10 | 14.40 | 5.50 | 100.00% | 100.00% |
| Predicted | 611.95 | | 0.09 | 13.33 | 6.58 | 99.99% | 99.99% |
| Error | 2.00% | | 10.00% | 7.43% | 19.64% | 0.01% | 0.01% |
| Experiment 9.2 | | | | | | | |
| Measured | 154.59 | N/A | 0.05 | 4.10 | 5.85 | 99.70% | 100.00% |
| Predicted | 198.16 | | 0.09 | 3.89 | 6.02 | 96.46% | 96.46% |
| Error | 28.18% | | 80.00% | 5.12% | 2.91% | 3.25% | 3.54% |

Table X

Experiment 10 Results

| | Mean Response Time | | Mean Number in Queue | | | Utilization | |
|---|---|---|---|---|---|---|---|
| | Class 1 | Class 2 | Disk I/O | Run | TTY I/O | CPU0 | CPU1 |
| Experiment 10.1 | | | | | | | |
| Measured | 275.07 | 405.28 | 0.40 | 11.10 | 8.50 | 100.00% | 100.00% |
| Predicted | 219.91 | 393.72 | 0.43 | 9.47 | 10.09 | 99.87% | 99.87% |
| Error | 20.05% | 2.85% | 7.50% | 14.68% | 18.71% | 0.13% | 0.13% |
| Experiment 10.2 | | | | | | | |
| Measured | 173.98 | 255.11 | 0.15 | 7.30 | 8.55 | 100.00% | 100.00% |
| Predicted | 149.45 | 255.19 | 0.39 | 5.94 | 9.66 | 98.64% | 98.64% |
| Error | 14.10% | 0.03% | 160.00% | 18.63% | 12.98% | 1.36% | 1.36% |
| Experiment 10.3 | | | | | | | |
| Measured | 75.80 | 110.32 | 0.10 | 3.20 | 6.70 | 73.40% | 89.60% |
| Predicted | 81.65 | 129.90 | 0.26 | 2.32 | 7.42 | 87.27% | 87.27% |
| Error | 7.72% | 17.75% | 160.00% | 27.50% | 10.75% | 18.90% | 2.60% |

Table XI

Experiment 11 Results

| | Mean Response Time | | Mean Number in Queue | | | Utilization | |
|---|---|---|---|---|---|---|---|
| | Class 1 | Class 2 | Disk I/O | Run | TTY I/O | CPU0 | CPU1 |
| **Experiment 11.1** | | | | | | | |
| Measured | 202.09 | 442.25 | 0.30 | 10.80 | 8.90 | 100.00% | 100.00% |
| Predicted | 173.21 | 407.32 | 0.47 | 8.94 | 10.58 | 99.82% | 99.82% |
| Error | 14.29% | 7.90% | 56.67% | 17.22% | 18.88% | 0.18% | 0.18% |
| **Experiment 11.2** | | | | | | | |
| Measured | 115.50 | 277.92 | 0.15 | 7.00 | 8.85 | 100.00% | 100.00% |
| Predicted | 119.77 | 265.19 | 0.42 | 5.62 | 9.96 | 98.45% | 98.45% |
| Error | 3.70% | 4.58% | 180.00% | 19.71% | 12.54% | 1.55% | 1.55% |
| **Experiment 11.3** | | | | | | | |
| Measured | 55.17 | 113.87 | 0.10 | 2.80 | 7.10 | 75.80% | 93.00% |
| Predicted | 68.05 | 138.57 | 0.27 | 2.23 | 7.50 | 86.85% | 86.85% |
| Error | 23.35% | 21.69% | 170.00% | 20.36% | 5.63% | 14.48% | 6.61% |

### V.2.6 Experiment 12

The results of experiment 12 are shown in Table XII. The workloads for this experiment are the same as used for experiment 6. For this experiment, the one processor node representation was again at least as accurate as the model with two processor nodes.

## V.3 Application of Service Time Adjustment

In Chapter III, the problem of unequal service requirements for a Round-Robin queueing discipline was discussed. Eq (14) was presented as a method for adjusting the service times to compensate for differences between the real world system and the assumptions imposed by the model.

The application of Eq (14) requires that at least one model run be made without the service time adjustments. This initial model run permits the CPU utilization to be predicted. The service time of each class of customer is then divided into the portions of service received while in PQ1 and for each cycle through the Round-Robin scheduling of PQ2. Adjustments are made to each portion of service received in PQ2 and the new service time requirements determined by adding the individual parts. Additional service also is determined for the overhead of the Operating System. Once computed, the adjusted service time requirements can be applied to a model. This procedure was applied to the model and workloads used for experiments 1

Table XII

Experiment 12 Results

| | Mean Response Time | | Mean Number in Queue | | | Utilization | |
|---|---|---|---|---|---|---|---|
| | Class 1 | Class 2 | Disk I/O | Run | TTY I/O | CPU0 | CPU1 |
| Experiment 12.1 | | | | | | | |
| Measured | 86.18 | 27.15 | 0.70 | 3.60 | 15.70 | 69.40% | 84.20% |
| Predicted | 82.11 | 19.61 | 0.80 | 1.97 | 17.24 | 80.44% | 80.44% |
| Error | 4.72% | 27.77% | 14.29% | 45.28% | 9.81% | 15.91% | 4.47% |
| Experiment 12.2 | | | | | | | |
| Measured | 76.55 | 22.69 | 0.15 | 2.60 | 13.25 | 66.00% | 71.50% |
| Predicted | 70.56 | 16.75 | 0.58 | 1.37 | 14.05 | 71.21% | 71.21% |
| Error | 7.82% | 26.18% | 286.67% | 47.31% | 6.04% | 7.89% | 0.41% |
| Experiment 12.3 | | | | | | | |
| Measured | 59.77 | 21.83 | 0.70 | 1.70 | 7.60 | 44.10% | 38.20% |
| Predicted | 59.55 | 13.95 | 0.31 | 0.74 | 8.95 | 52.37% | 52.37% |
| Error | 0.37% | 36.10% | 55.71% | 56.47% | 17.76% | 18.75% | 37.09% |

through 6. The service times and transition probabilities in Figure 9 are those used for experiment 18. The results of this series of experiments are found in Table XIII through XVIII.

For most applications of the service time adjustment, the response time errors remained about the same or slightly higher than those of the corresponding experiments 1 through 6. However, the application of service time adjustments to the workload of experiment 4 caused the response time errors to decrease by a significant amount when the processors were heavily utilized. These results appear in Table XVI. The service time requirement of the jobs in that experiment were such that that the last cycle through the Round-Robin PQ2 queue required a small amount of service when compared with the full quantum allocated to the jobs in the queue. As a result, jobs on the real world system had to wait in the queue to receive a small amount of service from the processors. The use of the service time adjustment is an attempt to correct for this additional wait not incorporated in the model. The procedure shows good improvement in response time errors for jobs which had a small service time requirement for the final pass through the Round-Robin cycle of PQ2. There is some negative impact on other job mixes but the effect on the workloads used in this report were relatively small.

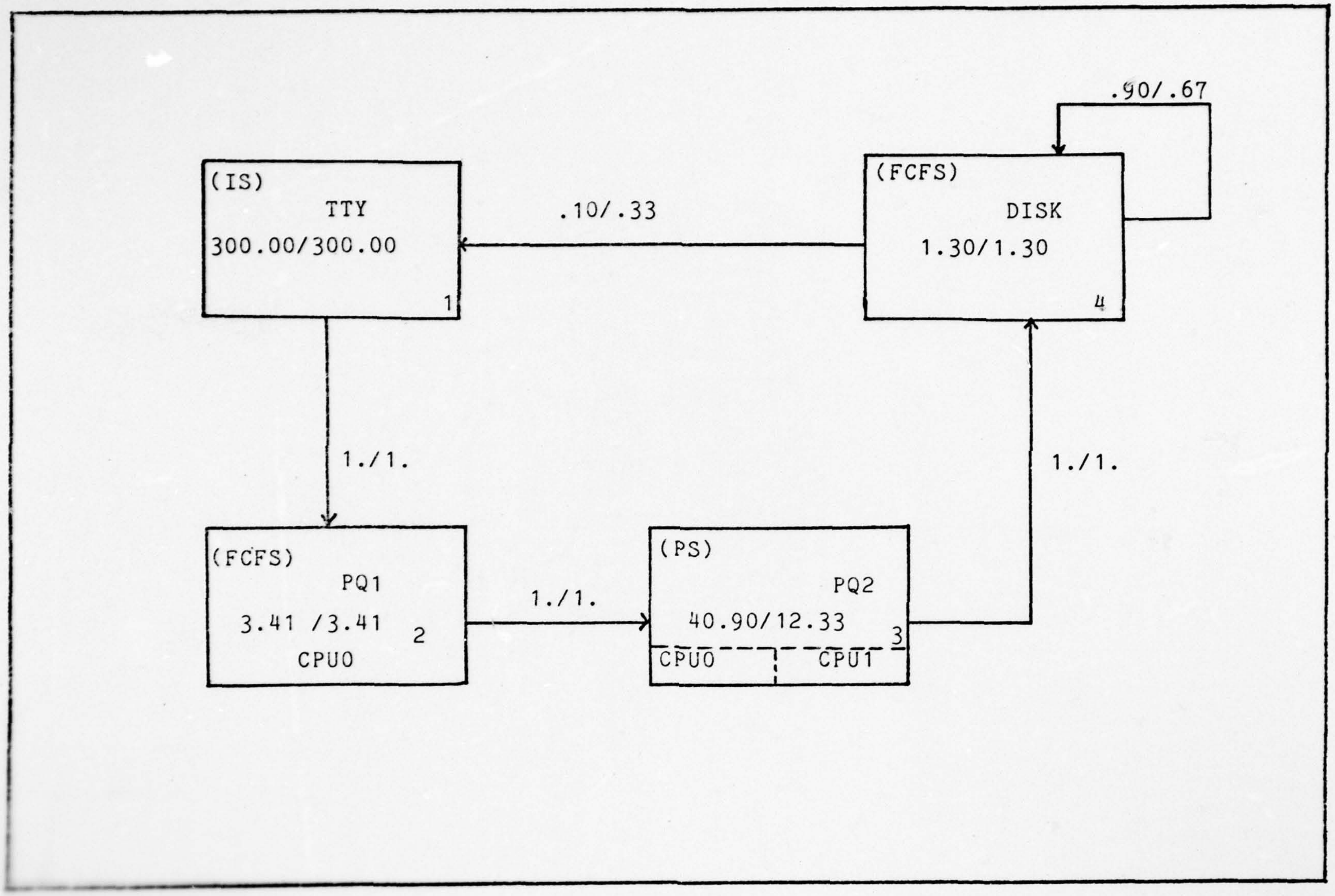


Figure 9. Service Time Adjustment Model

AD-A055 118   AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO   SCH--ETC   F/G 9/2
VALIDATION OF CLOSED QUEUEING NETWORK MODELS FOR THE DECSYSTEM---ETC(U)
MAR 78   H E SAXTON

UNCLASSIFIED   GCS/EE/77-7   NL

2 OF 2
AD
A055 118

END
DATE
FILMED
7 -78
DDC

Table XIII

Experiment 13 Results

| | Mean Response Time | | Mean Number in Queue | | | Utilization | |
|---|---|---|---|---|---|---|---|
| | Class 1 | Class 2 | Disk I/O | Run | TTY I/O | CPU0 | CPU1 |
| Experiment 13.1 | | | | | | | |
| Measured | 14.72 | N/A | 0.40 | 1.50 | 18.10 | 40.40% | 30.40% |
| Predicted | 14.42 | | 0.33 | 0.59 | 19.08 | 26.97% | 26.97% |
| Error | 2.04% | | 17.50% | 60.67% | 5.41% | 33.24% | 11.28% |
| Experiment 13.2 | | | | | | | |
| Measured | 13.22 | N/A | 0.15 | 1.40 | 8.45 | 33.20% | 10.40% |
| Predicted | 13.10 | | 0.14 | 0.28 | 9.58 | 14.49% | 14.49% |
| Error | 0.91% | | 6.67% | 80.00% | 13.37% | 56.36% | 39.33% |

Table XIV

Experiment 14 Results

| | Mean Response Time | | Mean Number in Queue | | | Utilization | |
|---|---|---|---|---|---|---|---|
| | Class 1 | Class 2 | Disk I/O | Run | TTY I/O | CPU0 | CPU1 |
| Experiment 14.1 | | | | | | | |
| Measured | N/A | 105.26 | 0.85 | 4.60 | 14.55 | 97.10% | 98.80% |
| Predicted | | 91.60 | 1.67 | 3.01 | 15.32 | 86.91% | 86.91% |
| Error | | 12.98% | 96.47% | 34.57% | 5.29% | 10.49% | 12.03% |
| Experiment 14.2 | | | | | | | |
| Measured | N/A | 47.54 | 0.30 | 1.90 | 7.80 | 51.20% | 61.80% |
| Predicted | | 56.20 | 0.53 | 1.04 | 8.42 | 59.40% | 59.40% |
| Error | | 18.22% | 76.67% | 45.26% | 7.95% | 16.02% | 3.88% |

Table XV

Experiment 15 Results

| | Mean Response Time | | Mean Number in Queue | | | Utilization | |
|---|---|---|---|---|---|---|---|
| | Class 1 | Class 2 | Disk I/O | Run | TTY I/O | CPU0 | CPU1 |
| Experiment 15.1 | | | | | | | |
| Measured | 624.41 | N/A | 0.10 | 14.40 | 5.50 | 100.00% | 100.00% |
| Predicted | 591.19 | | 0.10 | 13.17 | 6.73 | 100.00% | 100.00% |
| Error | 5.32% | | 0.00% | 8.54% | 22.36% | 0.00% | 0.00% |
| Experiment 15.2 | | | | | | | |
| Measured | 154.59 | N/A | 0.05 | 4.10 | 5.85 | 99.70% | 100.00% |
| Predicted | 192.14 | | 0.08 | 3.82 | 6.10 | 95.95% | 95.95% |
| Error | 24.29% | | 60.00% | 6.83% | 4.27% | 3.76% | 4.05% |

Table XVI

Experiment 16 Results

| | Mean Response Time | | Mean Number in Queue | | | Utilization | |
|---|---|---|---|---|---|---|---|
| | Class 1 | Class 2 | Disk I/O | Run | TTY I/O | CPU0 | CPU1 |
| Experiment 16.1 | | | | | | | |
| Measured | 275.07 | 405.28 | 0.40 | 11.10 | 8.50 | 100.00% | 100.00% |
| Predicted | 270.05 | 427.57 | 0.39 | 10.23 | 9.38 | 99.92% | 99.92% |
| Error | 1.82% | 5.50% | 2.50% | 7.84% | 10.35% | 0.08% | 0.08% |
| Experiment 16.2 | | | | | | | |
| Measured | 173.98 | 255.11 | 0.15 | 7.30 | 8.55 | 100.00% | 100.00% |
| Predicted | 183.61 | 279.43 | 0.36 | 6.54 | 9.10 | 99.02% | 99.02% |
| Error | 5.54% | 9.53% | 140.00% | 10.41% | 6.43% | 0.98% | 0.98% |
| Experiment 16.3 | | | | | | | |
| Measured | 75.80 | 110.32 | 0.10 | 3.20 | 6.70 | 73.40% | 89.60% |
| Predicted | 95.93 | 137.90 | 0.26 | 2.53 | 7.21 | 88.38% | 88.38% |
| Error | 26.56% | 25.00% | 160.00% | 20.94% | 7.61% | 20.41% | 1.36% |

Table XVII

Experiment 17 Results

| | Mean Response Time | | Mean Number in Queue | | | Utilization | |
|---|---|---|---|---|---|---|---|
| | Class 1 | Class 2 | Disk I/O | Run | TTY I/O | CPU0 | CPU1 |
| Experiment 17.1 | | | | | | | |
| Measured | 202.09 | 442.25 | 0.30 | 10.80 | 8.90 | 100.00% | 100.00% |
| Predicted | 153.37 | 373.01 | 0.51 | 8.42 | 11.07 | 99.70% | 99.70% |
| Error | 24.11% | 15.66% | 70.00% | 22.04% | 24.38% | 0.18% | 0.18% |
| Experiment 17.2 | | | | | | | |
| Measured | 115.50 | 277.92 | 0.15 | 7.00 | 8.85 | 100.00% | 100.00% |
| Predicted | 107.79 | 244.46 | 0.44 | 5.27 | 10.29 | 97.83% | 97.83% |
| Error | 6.68% | 12.04% | 193.33% | 24.71% | 16.27% | 1.55% | 1.55% |
| Experiment 17.3 | | | | | | | |
| Measured | 55.17 | 113.87 | 0.10 | 2.80 | 7.10 | 75.80% | 93.00% |
| Predicted | 64.97 | 134.02 | 0.28 | 2.16 | 7.56 | 85.00% | 85.00% |
| Error | 17.76% | 17.70% | 180.00% | 22.86% | 6.48% | 12.14% | 8.60% |

Table XVIII

Experiment 18 Results

| | Mean Response Time | | Mean Number in Queue | | | Utilization | |
|---|---|---|---|---|---|---|---|
| | Class 1 | Class 2 | Disk I/O | Run | TTY I/O | CPU0 | CPU1 |
| Experiment 18.1 | | | | | | | |
| Measured | 86.18 | 27.15 | 0.70 | 3.60 | 15.70 | 69.40% | 84.20% |
| Predicted | 94.95 | 32.87 | 0.76 | 2.63 | 16.61 | 84.84% | 84.84% |
| Error | 10.18% | 21.07% | 8.57% | 26.94% | 5.80% | 22.25% | 0.76% |
| Experiment 18.2 | | | | | | | |
| Measured | 76.55 | 22.69 | 0.15 | 2.60 | 13.25 | 66.00% | 71.50% |
| Predicted | 77.86 | 25.37 | 0.56 | 1.71 | 13.73 | 74.23% | 74.23% |
| Error | 1.71% | 11.81% | 273.33% | 34.23% | 3.62% | 12.47% | 3.82% |
| Experiment 18.3 | | | | | | | |
| Measured | 59.77 | 21.83 | 0.70 | 1.70 | 7.60 | 44.10% | 38.20% |
| Predicted | 63.56 | 19.82 | 0.30 | 0.88 | 8.82 | 53.50% | 53.50% |
| Error | 6.34% | 9.21% | 57.14% | 48.24% | 16.05% | 21.32% | 40.05% |

## V.4 Single Processor Model

The service time adjustment procedure was also applied to a model representing the DECsystem-10 with only one processor. This representation is shown in Figure 10. The service times and transition probabilities in this figure are those used for experiment 20.

Time-sharing users may have only one CPU available to them if the secondary processor malfunctions or if the real time simulation programs are being run at high priority. Experiments 19 and 20 were conducted with the secondary CPU switched off-line. The model was evaluated once for the predetermined service time requirements, plus overhead, and again for the adjusted service time requirement, calculated with Eq (14). Experiment 19 considered a single class of customer while experiment 20 considered two customer classes.

### V.4.1 Experiment 19

The workload for experiment 19 considered a single class of 10 jobs with the following characteristics:

| | | |
|---|---|---|
| Think Time | = | 300.0 jiffies |
| Processor Service | = | 83.6 jiffies |
| Disk I/O Calls | = | 3 |

The results of the experiment are shown in Table XIX. Part 1 presents the results using the required processor service time plus the overhead correction. This direct application of the model to the single processor configuration produces

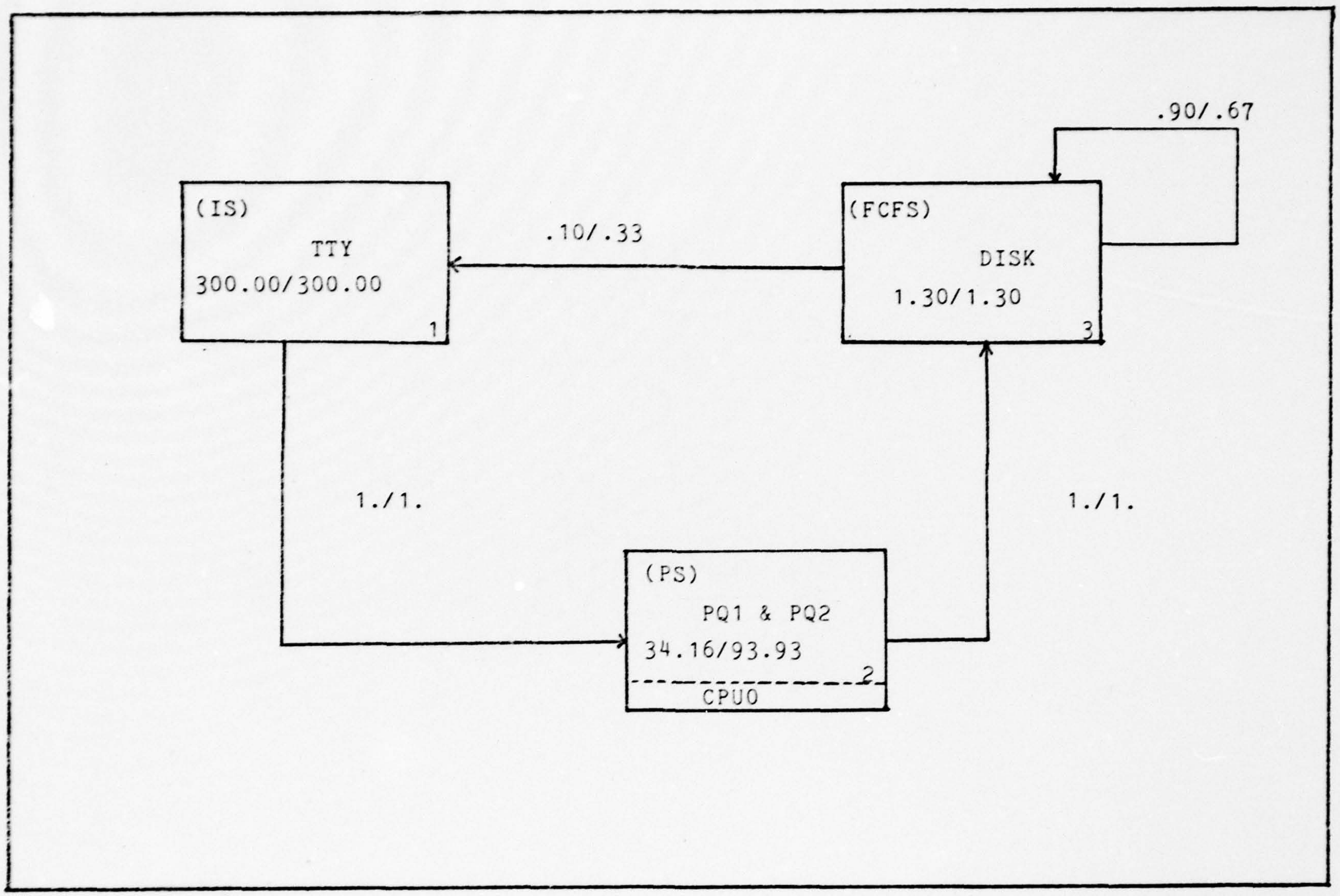


Figure 10. Single Processor Model

Table XIX

Experiment 19 Results

| | Mean Response Time | | Mean Number in Queue | | | Utilization | |
|---|---|---|---|---|---|---|---|
| | Class 1 | Class 2 | Disk I/O | Run | TTY I/O | CPU0 | CPU1 |
| Experiment 19.1 | | | | | | | |
| Measured | 629.33 | N/A | 0.05 | 7.70 | 2.25 | 100.00% | N/A |
| Predicted | 620.14 | | 0.04 | 6.69 | 3.26 | 99.84% | |
| Error | 1.46% | | 20.00% | 13.12% | 44.89% | 0.16% | |
| Experiment 19.2 | | | | | | | |
| Measured | 629.33 | N/A | 0.05 | 7.70 | 2.25 | 100.00% | N/A |
| Predicted | 633.18 | | 0.04 | 6.74 | 3.21 | 99.86% | |
| Error | 0.61% | | 20.00% | 12.47% | 42.67% | 0.14% | |

results remarkably close to measured values. This is partially accounted for by the relatively equal amounts of service required each time through the Round-Robin queue for PQ2. Each customer should receive two allocations of 30 jiffies and have a final requirement for 20.6 jiffies while in PQ2. However, since each job receives 3 jiffies of service while in PQ1, there should be some conflict for the use of the single processor on the real world system. In particular, even though jobs in PQ1 have priority over PQ2 jobs, the jobs entering PQ1 may still have to wait until the existing jobs at the resource completes its service quantum. Since the processor may be servicing a job from PQ2, the wait time could be as long as 30 jiffies. The jobs in the real world system could then be expected to experience additional waiting not included in the model solution. The small error value in this experiment may not be influenced so much by the accuracy of the model as by other offsetting factors, such as those to be discussed in experiment 20.

The results in part 2 were obtained after a correction was made for the differences in service requirements on different passes through the Round-Robin queue of PQ2. There was some improvement for nearly all the predicted values. Since the jobs in this experiment had a large remaining service requirement on the final pass through the Round-Robin queue, the service time adjustments were small, as indicated by the relatively small changes in the predicted values. The changes were however, in a direction which produced smaller errors.

## V.4.2 Experiment 20

The results for experiment 20 are shown in Table XX. The experiment used two 10 job classes with the following characteristics:

| | | |
|---|---|---|
| Think Time | | = 300.0 jiffies |
| Processor Service | | |
| | Class 1 = | 30.6 jiffies |
| | Class 2 = | 83.6 jiffies |
| Disk I/O Calls | | |
| | Class 1 = | 10 |
| | Class 2 = | 3 |

The results in part 1 of Table XX show that larger errors are produced when differing processor service time requirements are included. The response time errors would appear to indicate that class 1 jobs are experiencing some delay not anticipated by the model. This may be correct or some unexpected delay may be most evident on the class 1 jobs. Class 1 customers receive 27.6 jiffies of service during the pass through PQ2. However, it was noted that the jobs of both classes were occassionly requeued before receiving a full quantum of service. The data obtained from the METER program does not provide the reason for requeueing. However, the most likely cause is the expiration of the job's in-core protect time. When this occurs, the job is returned to the end of the queue even though it may not be necessary to swap it out of core. Since the jobs were requeued before receiving their full quantum, the expected amount of service required on the last pass through PQ2 was caused to vary. Analysis of several runs of METER data

Table XX

Experiment 20 Results

| | Mean Response Time | | Mean Number in Queue | | | Utilization | |
|---|---|---|---|---|---|---|---|
| | Class 1 | Class 2 | Disk I/O | Run | TTY I/O | CPU0 | CPU1 |
| Experiment 20.1 | | | | | | | |
| Measured | 642.28 | 1359.52 | 0.05 | 15.70 | 4.25 | 100.00% | N/A |
| Predicted | 507.30 | 1328.25 | 0.22 | 14.22 | 5.56 | 100.00% | |
| Error | 21.02% | 2.30% | 340.00% | 9.43% | 30.82% | 0.00% | |
| Experiment 20.2 | | | | | | | |
| Measured | 642.28 | 1359.52 | 0.05 | 15.70 | 4.25 | 100.00% | N/A |
| Predicted | 512.86 | 1351.27 | 0.22 | 14.27 | 5.51 | 100.00% | |
| Error | 20.15% | 0.61% | 340.00% | 9.11% | 29.65% | 0.00% | |

indicated that the jobs in this experiment received an average of 24 jiffies of service on the first pass through PQ2. This implies that class 1 customers could be expected to make an extra pass through the queue. The effect on class 2 customers is reduced since they require several passes in any event.

The results of the application of the service time adjustment for this experiment are shown in part 2 of Table XX. There is some improvement in the errors. However, the service time adjustment was based on jobs receiving an allocation of 30 jiffies in PQ2. If the jobs are requeued before completing service because of the expiration of in-core protect time, then a new adjustment is needed. In order to determine the new adjustment, it is necessary to know the amount of service received in each pass through the PQ2 queue. An average value was determined for this experiment but the location of the jobs within the run queue when in-core protect time expires will vary with the workload.

The results of experiments 19 and 20 indicate that Closed Queueing Network Models may be applied with reasonable accuracy to the DECsystem-10 in the single processor configuration. The increased response time highlighted another problem in applying this class of model to complex computer systems. The expiration of the in-core protect time tended to have a randomizing effect on the amount of service received during the first pass through the Round-Robin queue of PQ2. Certain classes of customers were more sensitive than others to this change. Since the amount of time spent in the run queue will be partially dependent upon the workload, no fixed service adjustment is possible. In order to fully analyze this problem a more detailed study of the scheduling and swapping algorithms is needed.

## V.5 Customer Swapping Model

A Closed Queueing Network Model was used to represent the DECsystem-10 when customer swapping was required. This model is shown in Figure 11. Swapping is required when the total memory requirement of all users exceeds the core capacity of the system. Since the addition of 512K words of core memory, excessive swapping has not been apparent during operation of the DECsystem-10 at AFAL. However, user requirements often tend to expand to the capacity of critical resources such as memory. For this reason, a workload was developed which induced swapping so that the TRACK and METER programs could be used to analyze the effects on the system performance.

### V.5.1 Experiment 21

The workload for experiment 21 was comprised of two classes of 10 jobs each, with the following characteristics:

```
Think Time         = 300.0 jiffies
Processor Service
            Class 1 =  30.6 jiffies
            Class 2 =  83.6 jiffies
Disk I/O Calls
            Class 1 =  10
            Class 2 =   3
```

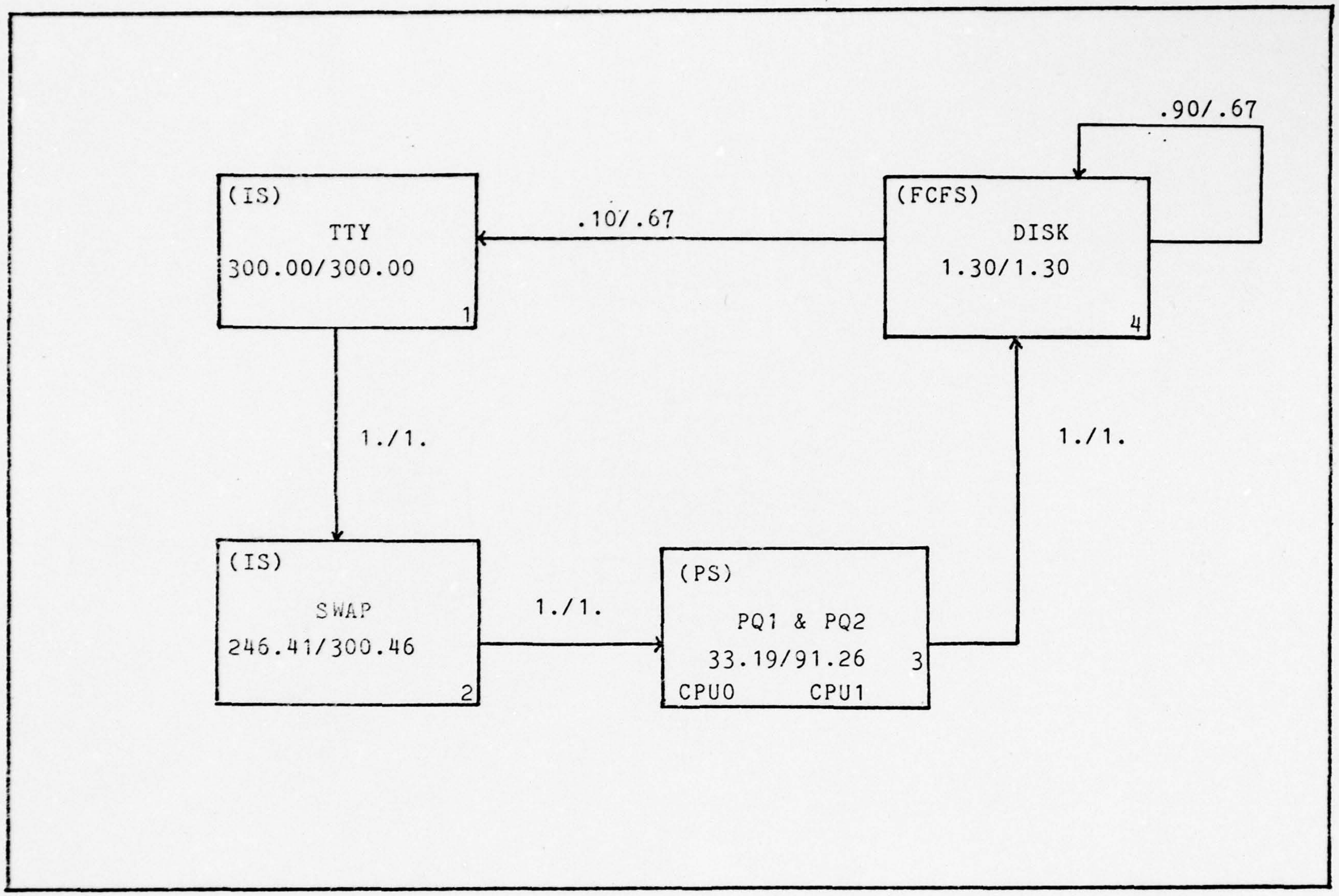


Figure 11. Customer Swapping Model

In addition to the above characteristics, each job was allocated 50K words of core. The workload of 10 jobs in class 1 and 10 jobs in class 2 required a total of 1000K words of core. The SYSTAT program indicated that available user memory could only contain 13 of these jobs at one time. The remaining 7 would have to be swapped onto a specified RP04 disk unit.

The additional load on the Operating System imposed by job swapping caused a large increase in measured processor overhead. The overhead of CPU1 increased to 9%, while the overhead of CPU0 was measured at 18%. The mean service time requirement of each class was calculated in the same manner as earlier experiments without the use of the service time adjustment. The average delay for swapping was calculated from the data of the METER program.

The results of this experiment are shown in Table XXI. Using the measured swapping delays produced good response time errors. However, the availability of the average swapping delay for general workloads is questionable. Neither the TRACK nor SYSTAT programs provided parameters which could be directly related to the measured values for swapping delay. The fact that 13 customers could occupy memory at one time does not imply that the Swapper Module does not attempt to optimize performance and swap out more jobs than than is actually required for this deterministic workload. In addition, some of the delay may be absorbed as the customer waits in the PQ1 queue. This

Table XXI

Experiment 21 Results

| | Mean Response Time | | Mean Number in Queue | | | Utilization | |
|---|---|---|---|---|---|---|---|
| | Class 1 | Class 2 | Disk I/O | Run | TTY I/O | CPU0 | CPU1 |
| Experiment 21.1 | | | | | | | |
| Measured | 473.77 | 316.03 | 0.50 | 13.60 | 5.90 | 100.00% | 98.80% |
| Predicted | 517.86 | 345.30 | 0.32 | 11.36 | 8.32 | 93.29% | 93.29% |
| Error | 9.31% | 9.26% | 36.00% | 16.47% | 41.02% | 6.71% | 5.58% |

queue can normally be expected to be small but its length will be dependent upon the workload.

Another factor to be considered when swapping occurs is the interaction between user disk I/O operations and the system swapping. A similar workload which required that the synthetic jobs write onto a RP04 disk showed a drastic increase in measured I/O time when swapping was required. In that instance, the Operating System and the user programs were both using the same disk control unit. The average user disk interaction was measured at 1.72 jiffies when the user I/O was to a RP03 disk unit. When swapping was induced and both the Operating System and the synthetic jobs used the RP04 disk units, the average user disk interaction time increased to approximately 40 jiffies. The size of the increase in I/O time warrants additional investigation in later studies.

The representation of the DECsystem-10 shown in Figure 11 has been shown to produce relatively good predictions of the system performance when swapping is required. However, the value of the model is dependent upon the availability of the values for average swapping delay for each customer.

# Chapter VI

# Conclusions and Recommendations for Future Study

## VI.1 Conclusions

The Closed Queueing Network Model provides a flexible means of representing complex computer systems. Several representations of the DECsystem-10 have been presented in this report. The analytic solution presented in Chapter III placed requirements on the system being modeled which are not totally satisfied by the DECsystem-10. In particular, the system does not normally satisfy the limitation on service rates for FCFS service centers and the requirement that a customer may only occupy one resource queue at a time. The validation of the different models used to represent the system has indicated how certain system features and parameters of the DECsystem-10 affect the accuracy of various model representations.

The dual processor node model used for experiments 1 through 6 produced response time errors ranging up to 33%. Although the accuracy is not as good as would be desired for detailed performance evaluation of the system, the flexibility of Closed Queueing Network Models permits many configurations to be easily represented. The alternate representation used for experiments 7 through 12 indicated that equally accurate performance measures may be obtained with a simplified model using only one Processor Sharing node to represent the processors. Finally, adjustments were made to the service time requirements of the customers in an effort to compensate for the limitations imposed by the closed form solution of the model. The results from the application of the adjusted service times, experiments 13 through 18, indicated that the errors for worst case workloads are improved while there is a slight decrease in accuracy for other workloads.

A Closed Queueing Network Model was also used to represent the DECsystem-10 in a single processor configuration. The response time errors were relatively small with the exception of class 2 users in experiment 20. That experiment highlighted the influence of the in-core protect time counter. Even though swapping was not required for the workload in that experiment, the in-core protect time had a distinct effect on the system performance for specific customer classes. Although additional investigation is needed in this area, the model used for this configuration provides a flexible representation of the system and yields results of reasonable accuracy.

Finally, the effects of swapping on system performance were investigated. The results of experiment 21 indicate that a delay node can be used to represent the swapping delay. However, the application of the model requires that swapping times be taken from METER or equivalent data. The collection of this detailed information is generally not practical during normal operation. Even though the response time errors were below 10%, the requirement for the average swapping delay makes the application of this model impractical at the present time. Further study may determine a correlation between the average swapping delay and one or more parameters available in the TRACK data. A relationship of this form would permit the application of the model to the DECsystem-10.

Closed Queueing Network Models offer great flexibility for representing complex computer systems such as the DECsystem-10. The product form solution to this class of model requires that assumptions be made concerning the system to be modeled. This report has shown that the product form solution for Closed Queueing Network Models is applicable to the DECsystem-10 if a high degree of accuracy is not required.

## VI.2 Recommendations for Future Study

The magnitude of the response time errors for the models presented in this report arise from two areas; limitations in the model caused by assumptions made to obtain a closed form solution and limitations in the construction of the synthetic workload. Additional topics for future study include:

1. The limitations imposed by the assumptions required for the model's solution need to be further

investigated. A rigorous mathematical investigation should be undertaken in order to determine refined service time adjustment procedures and the limitations of such procedures.

2. A synthetic workload which more closely approximates the assumptions of the model needs to be developed in order to effectively validate models of this nature. The SCRIPT program permits the simultaneous execution of numerous programs. Classes could be modeled by several synthetic jobs loaded into the system using several "scripts" for each class of customer.

3. A more comprehensive investigation of scheduling and swapping procedures is required to analyze the problem of early requeueing of jobs in the run queue. In addition, a sensitivity study should be performed to determine the effects on performance resulting from changes in the resource allocation parameters.

4. Additional study of I/O features needs to be accomplished to validate this class of model for the complete system. This investigation should include the consequences of swapping on I/O and overall system performance.

5. In order for a model which incorporates customer swapping to be of practical value, readily accessible parameters which provide good correlation with observed swapping delays must be identified. The identification of these parameters will require detailed knowledge of the Operating System Swapping routines.

## References

1. McKenzie, L. E. Queueing Network Model for Performance Evaluation of the DECsystem-10. AFIT-GCS-EE-77-1. Wright Patterson Air Force Base, Ohio; AFIT, March 1977.

2. Data Research Corporation. "Digital Equipment DECsystem-10." Datapro: 70C-384-01a-70c-384-03b. Delran, New Jersey: Datapro Research Corporation, June 1976.

3. Ampex Corporation. Technical Manual for Ampex Replacement Memory, Model ARM-10L. El Segundo, California: Ampex Corporation, 1976.

4. Digital Equipment Corporation. DECsystem-10 Monitor Structure. Unpublished Course Handout. Maynard, Massachusetts: Digital Equipment Corporation, (No date available).

5. Hellerman, H. and T. F. Conroy. Computer System Performance. New York: McGraw-Hill Book Co., 1975.

6. Svobodova, L. Computer Performance Measurement and Evaluation Methods: Analysis and Application. New York: American Elsevier Publishing Co.,1976.

7. Smolsky, E., et. al. DECsystem-10 Program Logic Manual for Scheduler and Swapper. Kalamazoo, Michigan; Western Michigan University, May 1977.

8. Kleinrock, L. Queueing Systems, Volume II: Computer Applications. New York: John Wiley and Sons, Inc., 1976.

9. Digital Equipment Corporation. DECsystem-10 Monitor Calls Manual. Maynard, Massachusetts: Digital Equipment Corporation, 1974.

10. Digital Equipment Corporation. DECsystem-10 Operating System Commands Manual. Maynard, Massachusetts: Digital Equipment Corporation, May 1977.

11. Baskett, F., et. al. "Open, Closed and Mixed Networks of Queues with Different Classes of Customers." Journal of the ACM, 22:248-260 (April 1975).

12. Jackson, J. R. "Jobshop-Like Queueing Systems." Management Science, 10:131-142 (October 1963).

13. Gordon, W. J. and G. F. Newell. "Closed Queueing Systems with Exponential Servers." Operations Research, 15:254-265. (March 1967).

14. Moore, C. G. Network Models for Large-Scale Time-Sharing Systems. Technical Report 71-1. Ann Arbor, Michigan; University of Michigan, April 1971. AD 727206

15. Cox, D. R. "A Use of Complex Probabilities in the Theory of Stochastic Process." Proceedings of the Cambridge Philosophical Society, 51:313-319 (1955).

16. Chandy, K. M. "The Analysis and Solutions for General Queueing Networks." Proceedings Sixth Annual Princeton Conference on Information Science and Systems: 224-228. Prince University, March 1972.

17. Wong, J. W. Queueing Network Models for Computer Systems. UCLA-ENG-7579 Los Angeles, California; UCLA, October 1975.

18. Chen, P. P. "Queueing Network Model of Interactive Computing Systems." Proceedings of the IEEE, 63:954-957 (June 1975).

19. Little, J. D. "A Proof of the Queueing Formula $L=\lambda W$." Operations Research, 9:383-387 (May 1961).

20. Reiser, M. "Interactive Modeling of Computer Systems." IBM System Journal, 4:309-327 (1976).

21. Giammo, T. "Validation of a Computer Performance Model of the Exponential Queueing Network Family." Acta Informatica, 7:137-152 (1976).

22. Buzen, J. P. Queueing Network Models of Multiprogramming, Ph.D. Thesis, Division of Engineering and Applied Science, Harvard University, Cambridge, Massachusetts, 1971. AD 731575

23. Cox, D. R. and W. L. Smith Queues. New York: John Wiley and Sons, Inc., 1961.

# Appendix A

## Closed Queueing Network Model Solution Technique

Networks of queues provide an flexible method of modeling multiprogrammed and time-sharing computer systems. A computer system may be viewed as a network of resources (terminals, CPUs, I/O devices) and a collection of customers and their associated tasks. Similar resources may be grouped together into a single class called a service center. The customers pass from one service center to another, receiving some amount of service from each of the centers along their route. If the routes are such that a customer may neither enter not leave the network, then the network is said to be a closed queueing network. In a time-sharing environment, one of the service centers will represent the user's terminals. The command entered by the user proceeds to the system resources where it receives some amount of service and returns to the terminal center where it receives more service in the form of think time. Following the think time, another task leaves the terminal service center and proceeds through the network. The closed queueing network then provides a realistic model of a time-sharing computer system.

Associated with each service center in the network is a queue. Customers are assigned to a service center's queue while they are awaiting or receiving service from that center. The state of a closed queueing network can then be described in terms of the number of customers at each queue in the network.

The system changes state when a customer departs one service center and enters the queue of another. Jackson (Ref 12) and Gordon and Newell (REF 13) have described the long-run-average behavior in terms of an equilibrium state probability distribution function. At equilibrium, the rate of transition of customers into a state from all other states must equal the rate of transition of customers out of that state into all other states. The equations describing this feature are called global balance equations by Chandy (Ref 16). Gordon and Newell used this concept when solving the balance equations for the case of identical customers. Customers are said to be identical if their routing through the network is the same and if their service time distribution functions are the same at each service center. The following discussion covers the development of a closed form solution for the equilibrium state probability distribution (Ref 13).

While deriving the steady state probabilities, two additional functions will be helpful. Let

$$\varepsilon(n_i) = \begin{cases} 0 & n_i = 0 \\ 1 & n_i > 0 \end{cases} \qquad (26)$$

This function is a dimensionless, multiplicative factor which will be used to eliminate from the steady state probabilities terms which cannot exist. Thus, the final result will be simplified with the understanding that the expression is valid only for

$$n_i \geq 0 \quad \text{and} \quad \sum_{i=1}^{M} n_i = N$$

Another function which will simplify notation is:

$$a_i(n_i) = \begin{cases} n_i & n_i < r \\ r & n_i > r \end{cases} \qquad (27)$$

This function is also a dimensionless, multiplicative factor. It will be used to indicate the number of jobs receiving service at the ith service center, with r identical servers, when there are $n_i$ jobs in the queue. Actually the factor may be any positive function, but the use of this particular $a_i(n_i)$ will clarify its use in the development of the steady state probability distribution.

The network will be characterized by the following parameters:

N = the number of customers (tasks or jobs) receiving service in the network.

$p_{ij}$ = the probability that a customer leaving the ith service center will proceed to the jth service center.

$M$ = the number of service centers in the network.

$r_i$ = the number of identical servers at the ith service center.

$\mu_i$ = the departure rate from a server in the ith service center. Expressed in jobs/sec.

$\lambda_i$ = the arrival rate at the ith service center. As with the departure rate, expressed in jobs/sec.

$P(n_1,n_2,\ldots,n_M)$ = the joint steady state probability that there are $n_i$ customers at the ith service center. These probabilities should only be defined for $\sum n_i = N$.

Consider the rate of transition out of a state $(n_1,n_2,\ldots.n_M)$:

$$\sum_{k=1}^{M} \varepsilon(n_k)a_k(n_k)\mu_k P(n_1,n_2,\ldots,n_M) \tag{28}$$

This formula indicates that customers exit from state $(n_1,n_2,\ldots n_M)$ through the kth center as long as there is at least one customer present at that center. If there are no customers at the kth center, no departures are possible and the function $\varepsilon(n_k)$ will set the corresponding term equal to zero for the summation.

The rate of transition into a state $(n_1,n_2,\ldots n_M)$ will be expressed as:

$$\sum_{i=1}^{M} \sum_{\substack{k=1 \\ k\neq i}}^{M} \varepsilon(n_k) a_i(n_i+1) \mu_i p_{ik} \; P(n_1,..,n_k-1,..,n_i+1,..n_M)$$

$$+ \sum_{i=1}^{M} \varepsilon(n_i) a_i(n_i) \mu_i p_{ii} \; P(n_1,n_2,...,n_M) \qquad (29)$$

This expression indicates that a transition into a state $(n_1,n_2,...n_M)$ occurs from state $(n_1,....n_k-1,..n_i+1,..n_M)$ whenever a customer completes service at the ith center and then proceeds to the kth center. When $n_k = 0$, the state $(n_1,...n_k-1,...n_i+1,...n_M)$ cannot exist. However, for mathematical convenience, $P(n_1,...n_k-1,..n_i+1,..n_M)$ will be allowed to exist and $\varepsilon(n_k)$ will eliminate that term from the summation (Ref 22;220)

The queue dependent function $a_k(n_k)$ must be handled differently when i = k. In this case, a customer departs the ith service center and immediately returns to that center. Therefore, the queue dependent departure rate must be based on $n_i$ customers rather than $n_i+1$ customers.

At equilibrium, the rate of transition out of any state is equal to the rate of transition into that state. Therefore, the steady state probabilities must satisfy:

$$\sum_{k=1}^{M} \varepsilon(n_k) a_k(n_k) \mu_k P(n_1, n_2, \ldots, n_M)$$

$$= \sum_{i=1}^{M} \sum_{\substack{k=1 \\ k \neq i}}^{M} \varepsilon(n_k) a_i(n_i+1) \mu_i p_{ik} \; P(n_1, \ldots, n_k - 1, \ldots, n_i + 1, \ldots, n_M)$$

$$+ \sum_{i=1}^{M} \varepsilon(n_i) a_i(n_i) \mu_i p_{ii} \; P(n_1, n_2, \ldots, n_M) \qquad (30)$$

Now consider the equilibrium condition for a particular service center:

Number of Jobs Departing = Number of Jobs Arriving

Note that jobs may only depart from a service center if the center is being utilized and that it may be utilized only when at least one job is present in its queue. The expression for the left side of the equation is $\mu_i$ * (Active Time), while the right hand side is $\lambda_i$ * (Total Time). Reordering terms yields the definition for resource utilization:

Percent Utilization = (Active Time)/(Total Time) = $\lambda_i$ / $\mu_i$

This ratio is also called the traffic intensity and is sometimes given units of erlangs in honor of A. K. Erlang, a pioneer in congestion theory (Ref 23;40). The $\mu_i$ are parameters of the queueing network and will remain constant. The $\lambda_i$ will vary with the workload N. It is assumed that changes in the number of jobs in the network will not affect the relative utilizations of the service centers. This is intuitively acceptable when all the customers are identical. However when customers have differing service requirements at a First Come First Serve queue, then the assumption is no longer valid; as is discussed in Chapter III.

For centers with identical servers, the utilization is:

$$\frac{X_i}{a_i(n_i)} = \frac{\lambda_i/\mu_i}{a_i(n_i)} \tag{31}$$

Where $X_i$ is the utilization of an equivalent single server center. Both $X_i$ and $\lambda_i$ are functions of the workload, N. When the service times are state dependent and the function $a_i(n_i)$ is an arbitrary positive function, the utilization of the servers at the ith center is no longer expressed by Eq (31). In this instance, the concept of an individual server within the service center is no longer clearly defined.

In order to solve the balance equations, the method of seperation of variables will be used. As stated earlier, it is assumed that the utilizations of the service centers are functions of the network parameters $p_{ij}$ and $\mu_i$, and of the number of customers in the network. Once an equilibrium condition is reached, the $X_i$ remain constant. These utilizations are then used to define a new function as follows:

$$Q(n_1,n_2,\ldots,n_M) = \frac{1}{C(N)} \prod_{i=1}^{M} (X_i)^{n_i} \tag{32}$$

Where C is a normalization constant which will later be used to adjust for the dependence of the $X_i$ upon the number of customers in the network.

For notational convenience also define:

$$B_k(0) = 1$$

$$B_k(n) = a_k(n)\, B_k(n-1) \qquad \begin{array}{l} k=1,2\ldots M \\ n=1,2\ldots N \end{array} \tag{33}$$

These functions permit the following change of variables to be made in the balance equations:

$$P(n_1,n_2,\ldots,n_M) = \frac{Q(n_1,n_2,\ldots,n_M)}{\prod_{k=1}^{M} B_k(n_k)} \tag{34}$$

Note that

$$P(n_1,..n_k-1..n_i+1..n_M) = \frac{\frac{a_k(n_k)}{a_i(n_i+1)} Q(n_1,..n_k-1..n_i+1..n_M)}{\prod_{j=1}^{M} B_j(n_j)} \quad (35)$$

Substituting into the balance equations and canceling the terms yields the following results:

$$\sum_{k=1}^{M} \varepsilon(n_k) a_k(n_k) \mu_k Q(n_1,n_2,...n_M)$$

$$= \sum_{i=1}^{M} \sum_{k=1}^{M} \varepsilon(n_k) a_i(n_i) \mu_i p_{ii} \; Q(n_1,..n_k-1,..n_i+1,..n_M) \quad (36)$$

substituting for Q(·) yields:

$$\sum_{k=1}^{M} \varepsilon(n_k) a_k(n_k) \mu_k \left[\prod_{i=1}^{M} X_i^{n_i}\right]$$

$$= \sum_{i=1}^{M} \sum_{k=1}^{M} \varepsilon(n_k) a_i(n_i) \mu_i p_{ik} \; (X_i/X_k) \left[\prod_{j=1}^{M} X_j^{n_j}\right] \quad (37)$$

which reduces to:

$$\sum_{k=1}^{M} \varepsilon(n_k) a_k(n_k) \mu_k - \sum_{i=1}^{M} \sum_{k=1}^{M} \varepsilon(n_k) \mu_i p_{ik} \; (X_i/X_k) = 0 \quad (38)$$

Since all the jobs could be at any of the service centers at one time, the following must hold:

$$\sum_{i=1}^{M} \mu_i X_i p_{ik} = {}_k X_k \qquad k=1,2,\ldots,M \qquad (39)$$

From the earlier definition of $X_i$, this can be written as:

$$\sum_{i=1}^{M} \lambda_i p_{ik} = \lambda_k \qquad k=1,2,\ldots,M \qquad (40)$$

This is not an independent set of equations. The $\lambda_k$ can only be determined within a multiplicative constant. The interpretation of the set of equations is that the arrival rate to the kth center is equal to the weighted sum of the arrival rates of the service centers which could possibly send it a customer.

If the network is modeling a time-sharing computer system, then the relative arrival rates will be of interest. Let the user terminals be represented as workcenter number one. Then $\lambda_i / \lambda_1$ is the relative arrival rate at service center i with respect to the user terminals. This indicates that a customer will visit the ith service center $\lambda_i / \lambda_1$ times between the time it leaves the terminal until it returns.

Define $e_i$ as the expected number of visits to the ith center per interaction with the user terminal service center. This permits the last equation to be written as:

$$\sum_{i=1}^{M} e_i p_{ik} = e_k \qquad k=1,2,\ldots M \tag{41}$$

Now the relative number of visits to each center can be computed as well as the relative utilizations for identical servers, $X_i$. Rewriting the equilibrium state probability distribution function yields:

$$P(n_1, n_2, \ldots n_M) = \frac{1}{C(N)} \prod_{i=1}^{M} \frac{(X_i/\mu_i)^{n_i}}{B_i(n_i)} \tag{42}$$

Since the $X_i$ terms are utilizations, and the $B_i(n_i)$ terms are products of multiplicative factors, there are no units; as expected for a probability distribution. Note that:

$$B_i(n_i) = \prod_{k=1}^{n_i} a_i(k) \tag{43}$$

and let

$$\mu_j(k) = \mu_j a_j(k) \tag{44}$$

Where $\mu_j(k)$ is the departure rate from the jth center when there are k customers at the center. Then:

$$P(n_1, n_2, \ldots n_M) = \frac{1}{C(N)} F_1(n_1) F_2(n_2) \ldots F_M(n_M) \tag{45}$$

where

$$F_i(n_i) = \frac{e_i^{n_i}}{\prod_{j=1}^{n_i} \mu_i(j)} \qquad (46)$$

The $F_i(n_i)$ terms are unnormalized probabilities that service center i has $n_i$ customers in its queue.

The normalization constant is now written as C(N) to indicate its dependence upon the workload, N. C(N) must normalize the product on the right hand side of the equation to insure that the probabilities sum to one over all possible states. Therefore:

$$C(N)^{-1} = \sum_{\substack{\text{all states} \\ n_1+n_2+\ldots+n_M=N}} F_1(n_1)F_2(n_2)\ldots F_M(n_M) \qquad (47)$$

## Vita

Harold E. Saxton was born on 30 October 1945 in Indianapolis, Indiana. Following graduation from Palm Beach High School, he attended the University of Florida, Gainesville, Florida, where he received a Bachlor of Science degree in Mathematics. After graduation, he enlisted in the Air Force and received his commission through Officer Training School on 13 Aug 1969. His previous assignments include Tyndall AFB, Florida and Clark AB, Republic of the Philippines. Prior to his assignment to AFIT, he served as Radar Maintenance Officer at the 20th Surveillance Squadron, Eglin AFB, Florida.

Permanent address: 325 Executive Center Drive
Apt 113C
West Palm Beach, Florida
33401

**UNCLASSIFIED**

SECURITY CLASSIFICATION OF THIS PAGE (*When Data Entered*)

9/1

REPORT DOCUMENTATION PAGE — READ INSTRUCTIONS BEFORE COMPLETING FORM

2. GOVT ACCESSION NO. [redacted] 14 GCS/EE/77-7

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle) 6

VALIDATION OF CLOSED QUEUEING NETWORK MODELS FOR THE DECSYSTEM-10.

5. TYPE OF REPORT & PERIOD COVERED 9

[redacted] Master's thesis

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s) 10

Harold E./Saxton
[redacted]

8. CONTRACT OR GRANT NUMBER(s)

9. PERFORMING ORGANIZATION NAME AND ADDRESS

Air Force Institute of Technology (AFIT-EN)
Wright-Patterson AFB, Ohio 45433

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS

Advanced Systems Group (AFAL/AAF-2
Air Force Avionics Laboratory
Wright-Patterson AFB, Ohio 45433

12. REPORT DATE 11

Mar [redacted] 78

13. NUMBER OF PAGES 12

[redacted] 135 P.

14. MONITORING AGENCY NAME & ADDRESS (*if different from Controlling Office*)

15. SECURITY CLASS. (of this report)

UNCLASSIFIED

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (*of this Report*)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (*of the abstract entered in Block 20, if different from Report*)

18. SUPPLEMENTARY NOTES

Approved for public release; IAW AFR 190-17

[illegible]

JERRAL F. GUESS, Captain, USAF

19. KEY WORDS (*Continue on reverse side if necessary and identify by block number*)

Closed Queueing Network Models
Computer Performance Evaluation
Time-Sharing Computer Systems

20. ABSTRACT (*Continue on reverse side if necessary and identify by block number*)

A validation study was performed on Closed Queueing Network Models for the DECsystem-10 located at the Air Force Avionics Laboratory (AFAL). The purpose of this study was to provide the personnel of AFAL with an understanding of the accuracy with which this class of model can predict computer performance.

A previously developed Fortran program was used to implement the product form solution and the computational algorithms required for performance evaluation. A review of the solution technique is presented

→ next page

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

**UNCLASSIFIED**

SECURITY CLASSIFICATION OF THIS PAGE (*When Data Entered*)

∅12225

RC

with a discussion of the consequences resulting from assumptions required to arrive at the closed form solution.

The validation results are described for a series of experiments which investigated the accuracy of the models for several system configurations and under a variety of workloads. These results indicate that Closed Queueing Network Models are a flexible means of representing the DECsystem-10 and that the product form solution provides performance predictions which are useful for evaluation of the system.