THE UNIVERSITY OF MICHIGAN

COMPUTING RESEARCH LABORATORY

A STOCHASTIC MODEL OF MULTIPROCESSING

Brad Alan Makrucki

CRL-TR-23-84

MARCH 1984

Room 1079, East Engineering Building
Ann Arbor, Michigan 48109
USA
Tel: (313) 763-8000

A STOCHASTIC MODEL OF MULTIPROCESSING

by
Brad Alan Makrucki

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer, Information and Control Engineering)
in The University of Michigan
1984

Doctoral Committee:

Associate Professor Trevor N. Mudge, Chairman
Professor Daniel E. Atkins
Professor Gideon Frieder
Professor John P. Hayes
Professor Ronald J. Lomax

© Brad Alan Makrucki 1984
All Rights Reserved

ABSTRACT

A STOCHASTIC MODEL OF MULTIPROCESSING

by
Brad Alan Makrucki

Chairman: Trevor N. Mudge

A model of the behavior of multiprocessor systems consisting of processors, an interconnection network, and resource modules--typically memory devices--is developed. The model allows processor activity to be represented using stochastic finite state machines. These stochastic finite state machines may reflect program activity directly, or alternatively may reflect other levels of processor activity. The model is based on approximating processor behavior using semi-Markov processes, one for each processor. Using the semi-Markov descriptions, expressions are obtained for a set of performance measures that may be used to evaluate system performance at the level of system operation chosen for processor representation. The set of performance measures includes timing measures such as mean rates of computation, mean program execution times, and system component utilizations. The model is appropriate for more general systems where requestors, whose behavior may be modeled as semi-Markov processes, contend for system resources. In evaluating semi-Markov process quantities, queueing approximations are developed for waiting times in resource queues present within the

multiprocessor system. The queueing approximations also have applications in small finite-customer queueing network analysis. The model provides a framework and a set of analysis techniques that may be used in the development of models for specific applications. The model described unifies and extends previous models of multiprocessor resource contention by allowing more sophisticated descriptions of processor activity to be used.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE

LIST OF TABLES

NOTATION

If B is a random variable, it will be written with a˜ above it, such as $\tilde{B}$. $\tilde{B}$'s cumulative distribution function (CDF) will be written as $B(t) = Pr(\tilde{B} \leq t)$ while $\tilde{B}$'s probability density function (pdf) (assuming that it exists, otherwise use the Dirac delta function) will be written as $b(t) = \frac{dB(t)}{dt}$. Moments will be written as $\bar{B} = E[\tilde{B}]$, $\overline{B^k} = E[\tilde{B}^k]$. Important relationships (those critical to model solution) will be marked with a •. In diagrams or figures, dots (•) will be used to indicate global resource module reference states.

SUMMARY OF VARIABLES AND TERMINOLOGY

TERMINOLOGY

PE - processing element, typically a processor/requestor and some local hardware

GRM - global resource module, i.e., a system resource

ICN - interconnection network

SMP - semi-Markov process

ARP - alternating renewal process

VARIABLES

Variables post-fixed with a (p) indicate the quantity under consideration without PE p's contribution.

A_p - state space for SMP p

A_p^r - reduced state space for SMP p

$\tilde{B}(t)$ - bandwidth at time t

$\tilde{C}_{pij}$ - the state transition time for SMP p when moving from the entry into state i to the entry into state j

$\tilde{E}_{psm}$ - the excess connection time for the connection in progress when processor p emits a request for GRM m when it enters state s

$\bar{E}_{pm}$ - the mean excess connection time for processor p requesting GRM m, independent of the state of emission

f_{pm} - a multiplying factor involved in the computation of $\bar{E}_{pm}$

g_{pm} - a multiplying factor involved in the finite customer G/G/1 approximations

h_{pm} - a utilization factor used to approximate history effects

$\tilde{M}_p(n)$ - the embedded Markov chain state after n transitions in SMP p

$\tilde{N}_m(t)$ - the GRM m queue length at time t

$\mathbf{P}_p$ - the SMP p transition matrix

p_{ps} - the time limiting fraction of time that SMP p is in state s

$Q_p(i,j,t)$ - the semi-Markov kernel for SMP p

Q_{pm} - the set of requestors to be counted as in the queueing system associated with GRM m as seen by PE p

$\#_{pm}$ - the set of all PE's (not including p) that contribute work to GRM m

$\tilde{S}_{ps}$ - the sojourn time random variable for SMP p in state s

$\tilde{T}_p(i,\Omega)$ - the state to set transition time for SMP p. The time is measured from the exiting of state i to the entry to any state in set Ω.

u_{pm} - a usage factor used in the computation of f_{pm}

V_{M_m} - the coefficient of variation for the interarrival time of requests at GRM m

V_{P_p} - the coefficient of variation for the interemission of requests from PE p

$\tilde{W}_{psm}, \tilde{W}_{pm}$ - the waiting time seen by requests arriving at GRM m from p; $\tilde{W}_{psm}$ indicates the dependence on the state of emission while $\tilde{W}_{pm}$ ignores it.

$\tilde{X}_m$ - the connection time random variable for GRM m.

$\tilde{Y}_{psm}$ - the connection time random variable for PE p using GRM m when it is in state s

$\tilde{Z}_p(t)$ - the state of SMP p at time t

$\{\tilde{Z}_p(t), t \geq 0\}$ - the SMP describing PE p's behavior

α_{pm} - the probability that GRM m is is free at the PE p request arrival time

$\tilde{\Delta}_{pm}$ - the sequence of requests in the GRM m queue at the time of a request arrival from PE p

η_{psm} - the probability that PE p emits a request and it is for GRM m when it enters state s

Ω_{C_p} - the set of states that are computation states for PE p, $\Omega_{C_p} \subset A_p$

Ω_{R_p} - the set of states that are GRM reference states for PE p, $\Omega_{R_p} \subset A_p$

λ_{psm} - the rate of request flow from PE p in state s to GRM m

λ_{pm} - the rate of request flow from PE p to GRM m

λ_m^i - rate of request flow into GRM m

λ_p^o - rate of request flow out of PE p

ρ_m - GRM m utilization

ϕ_p - PE p utilization

π_p - the embedded Markov chain steady-state probabilities or fractions for SMP p

Equivalent Quantities

These quantities are associated with the ARP and are obtained in reducing PE SMP's to approximately equivalent ARP's, they are denoted with a $'$.

$\tilde{C}_p'$ - the equivalent ARP cycle time

$\tilde{S}_{j1}'$ - the equivalent ARP computation state sojourn time, or the "think time" associated with ARP p

$\tilde{Y}'_{pm}$ - the equivalent ARP p connection time for GRM m

η'_{pm} - the equivalent ARP probability that a request emitted by PE p is for GRM m

CHAPTER 1

INTRODUCTION

The decreasing cost of hardware has encouraged the introduction of multiprocessor computer systems. If carefully designed, multiprocessors can offer many desirable features, including high performance and fault-tolerance. However, there are many problems associated with multiprocessor design, operation, and programming. Design issues inherent in the specification of multiprocessor systems include: the choice of the number of processors in the system and their impact on performance; the configuration and operation of the interconnection network between processors and its effect on performance; and the structure of an operating system with primitives that control system resources and manage user programs.

To obtain information concerning these and other more specific design problems, system designers rely on a wide variety of techniques. Experience and intuition are often used as initial guidelines for making design choices. Simulation studies may then be used to verify or adjust design choices. However, simulation is often costly and rarely provides insight into underlying mechanisms. An alternative to simulation studies is the use of modeling as an evaluation tool. Model based evaluation of complex, real systems has both advantages and disadvantages.

Primary advantages of modeling compared to simulation include: the general understanding obtained from the model and its development; and the usually low cost incurred in solving model equations. The reward of general understanding should not be underestimated: a reasonably accurate model provides not only quantitative information but also a qualitative understanding of the system that is usually unobtainable from simulations. Simulation provides good estimates of system operating points if appropriate details are included but it does not describe the *relationship* between system parameters and performance measures unless an empirical model can be induced. This is practically impossible in most cases.

The main drawback of model based system evaluation is that of accuracy. Models are often predicated on assumptions that simplify mathematical analysis. Unfortunately, these assumptions often oversimplify and lead to inaccurate results. Without prior knowledge about the relative importance of various system parameters, it is imperative that accurate assumptions be used; as many real effects should be included in the model as are economically feasible. On the other hand, if a model is too detailed its solution cost may exceed the cost of simulation or direct system measurements. In addition, very detailed models may provide accurate solutions but often mask any qualitative understanding with their complexity.

Mean value models are quite convenient for obtaining rough estimates statements regarding qualitative aspects of system behavior. This knowledge suffices in many situations. It is our contention, though, that the most informative approach to modeling complex, real systems is to obtain results in the form of equations which are based on as few approximations as is practical. *Then simple, special case solutions may be pursued as desired by prospective model users.* It is our further contention that special case results are more useful when they have been derived from a more general model, because the effect of approximations can be estimated. Whitt describes a philosophy of advancing model development where heuristic approximations are used to encompass effects due to, for example, higher moments of random variables [Whi83]. That is, more sophisticated modeling is advocated beyond that often done presently. Although the context in [Whi83] is queueing network modeling, the same belief is shared here -- to describe system behavior more realistically than past models have done.

The framework and development of model relationships are emphasized here. The objective is to formulate a general model whose parameter structure forms a framework for future special case models to be developed. The model framework also allows extensions of the basic model to be developed in a straightforward manner.

To obtain general results which are not predicated on gross simplifying assumptions a notation must be developed to reflect the generalities. Occasionally, the notation may be cumbersome by comparison with models of a more simplistic nature -- this is due in part to the additional quantities that are not included in previous models. Merits of the increased completeness include:

(1) the ability to encompass *more performance measures within the model description of the system,* including estimates of program execution time

(2) the ability to develop more general relationships since the model assumptions are more realistic

1.1. Overview and Thesis

Figure 1.1 illustrates the logical system to which the model may be applied. Multiprocessor systems are typically composed of a set of processors which are connected together in the sense that they may communicate through some form of interconnection network. Of primary interest here are systems where processors are tightly coupled to form a local system, i.e., we are not explicitly considering computer networks. Rather, local multiprocessor systems are the direct application. Systems such as the Intel 432, the DEC System-10, C.mmp, and the IBM 360/158 fall roughly into this category.

Processors in the system are considered to be active devices that may originate system actions, e.g., event occurrences such as finishing jobs, executing instructions, etc. Processors may be grouped with local memory used for local data and instruction storage. Local cache memory may also be present in a processor group. This group of a processor with its local and cache memory along with any supplementary hardware will be termed a *processing element (PE).* PE's may behave autonomously in that they may execute programs stored in their local memory, alteratively they may have limited local computation capabilities such as I/O channels or special purpose processors.

Resources in the system are devices which respond to requests for service generated by PE's. Examples include: main memory modules; disk units; and some forms of special purpose processors that, for example, receive requests for service and respond with an acknowledge when done. Resources in the system will be termed *global resource modules (GRM's).* There may be several types of resources of the same or different sorts organized as modules which behave predominantly independently and may be accessed as global devices.

To complete the system an interconnection network through which service requests flow is required. Many possibilities exist in specifying the communication medium, they range from a set of buses which connect PE's and GRM's, to a full crossbar that allows all PE-GRM connection possibilities.

The *interconnection network (ICN)* considered by the model is shown in figure 1.1a. PE's are connected to one side of an ICN while resources are GRM's connected to the other side. This configuration might also appear as in figure 1.1b. The two configurations are not materially different, except that the model equations are presently written for the system organization shown in figure 1.1a. It is important to note that the model may be modified in appropriate places to allow modeling different configurations. For example, if the system is organized as in figure 1.1b, where clusters are is designed so that the GRM in each cluster gives preferential treatment to its local PE, model parameters may be specified, and equations may be re-written, to reflect the physical system specifications. Hence although the model applies to many systems, it is described in a specific context here.

In figure 1.1a there are *no* direct communication links between PE's. The model is primarily intended for MIMD (multiple instruction stream, multiple data stream) applications where dynamic (i.e., during system operation) coupling between PE's is small; for example, where PE's communicate through GRM's (not via low level synchronous communications). The basic model does not have the capability to accurately model synchronized PE communications or highly coupled PE behavior. There are approximate techniques discussed in chapter 8 that may be used to handle such cases but they have not been tested.

The model assumes that the ICN is virtually circuit switched. That is, each PE communicates with GRM's in a manner which *seems* to PE's to be circuit switched. The physical implementation is not considered explicitly although it may influence model parameters or require equation modifications. The virtual crossbar displays one level of connection delay and the total connection property (any PE-GRM connections not involving destination conflicts or broadcasting may exist simultaneously) with no internal queueing delay. This effect may be achieved with a very high speed bus or buses, or a crossbar -- which is becoming feasible with VLSI technology. Equations may be modi-

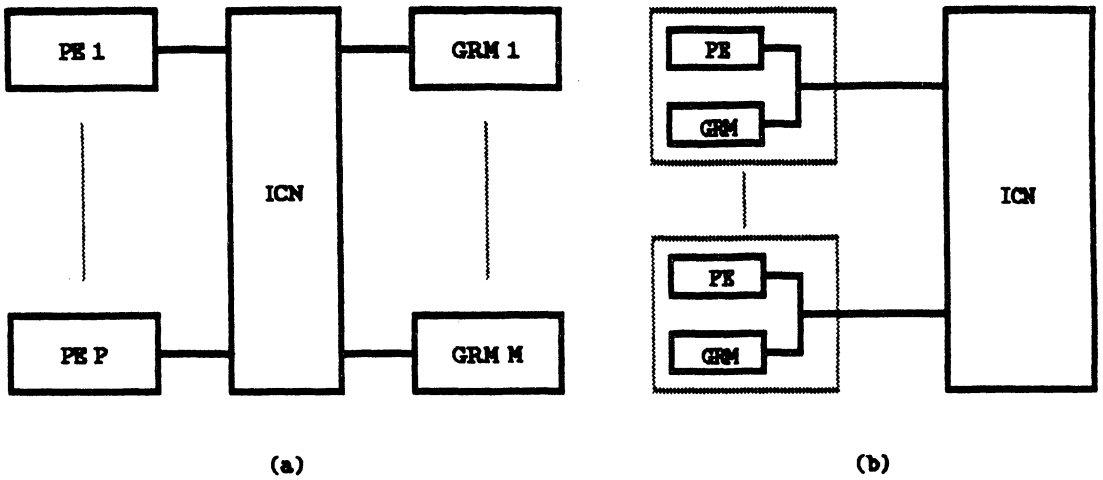


Figure 1.1 System configuration representation.

fied to include propagation delays.

Each PE executes a program which governs its activity. The term "program" will be used to describe the controlling aspect of a PE's behavior, whether the PE contains an instruction executing processor or not. In the case where PE's are, for example, I/O channels, the term "program" refers to the finite state machine that governs the channel's operational behavior. Programs executing on PE's may contend for service from the GRM's.

The model describes system behavior in terms of performance measures of the system. Hardware characteristics such as speed, and program characteristics such as branching fractions and timing specifications are parameters of the model.

Many relevant questions may be asked regarding system behavior at the level of description chosen. This dissertation attempts to quantify answers to questions such as:

1) At what rate is the system doing useful work, i.e. at what rate is it executing PE programs?

2) Is the system achieving its potential speed or is some speed (processing power) lost due to resource contention? How much is lost?

3) Is a particular piece of hardware being utilized efficiently or is it a bottleneck? If so, what is the bottleneck?

The model may be used as a comparison device for evaluating alternative system configurations, including programming. It does not explicitly guide the user to a better operating point.

More specifically, the model provides information on the following aspects of system operation: program state transition times or components thereof; PE utilizations and supplementary quantities that concern PE behavior; and GRM utilization and supplementary quantities.

State transition times are the times, or components of the times, required for a program to move from an initial state to a final state. This set of performance characteristics is important in determining the *rate* of program execution on a given PE and components of a program's execution time. An important aspect of the model development is the inclusion of enough generality that performance measures of sufficient utility may be predicted.

Consider timing aspects of program execution: programs execute up to a point where they need service from GRM's. At this point in PE program execution (all PE's are executing their own programs) the PE *emits a request* for service from an appropriate GRM. These requests are routed through the ICN to the requested GRM.

After requests reach their destination GRM's they enter GRM request queues where they are held until any service in progress has been completed (all GRM's operate "asynchronously"); consider a single GRM: at the completion of a service interval, the GRM controller examines the request queue for the next request for service. The queueing discipline is determined by GRM logic. Once a request from a PE reaches the GRM server (it may have been in the queue), the GRM controller transmits an acknowledge back to the originating PE informing the PE that it may begin use of the GRM.

Any modifications to the circuit switched property will require that equations describing the new situation be derived. Modifications in the network structure and behavior from above will impact the physical analysis of the resource subsystems, not the theory of the model.

The assumption of a virtual crossbar ICN is used because this work is primarily directed toward the modeling technique rather than the physical analysis of interconnection systems. The impact of this assumption will be noted where appropriate.

Notice that the circuit switched connection property is suitable for implementing data structure locks and semaphores in a very simple manner. For example, if a program needs to do a real/modify/write operation on the contents of a global memory location it would first obtain a connection to the appropriate GRM. Once this connection has been obtained, the PE is free to manipulate any contents of the GRM in an uninterruptible manner. This advantage should not be underestimated, it provides a very simple, clean implementation of what might otherwise be a special case architectural concern [GoG83].

After considering the behavior of PE's, GRM's and the ICN it becomes clear that programs that describe PE behavior may do so at many levels. The highest level of PE behavior description is that of an actual program flowchart. A PE description might alternatively be a low level PE description such as those that have been considered previously [BaS76, Bha75, Hoo77, MaG81, MuM82a, Pat79, Rau79, SeD79, SkA69, Smi74, Str70]. Early models have used statistics from program execution traces (such as GRM's referenced, inter-reference times, etc.) to provide parameter values for their equations. The model developed here uses program specifications as its set of parameters. The program need not necessarily have been compiled and executed for the model parameters to be deduced. The difference between the two description techniques is implied by the level of system description. In the early models, system activity was described at a level of processor timing without knowledge of high level processor activity such as the program it executes.

The level of PE description chosen during modeling depends on many factors such as the level and detail of the analysis results desired by the model user, and the amount of information available for the model being constructed.

With the generality offered by *the SMP model* (the model is based on using semi-Markov processes to describe PE activity) there may not be a unique choice for the PE activity description. For example, consider a simple example of an SMP model description of a system configuration (system and programs) under two conditions: where all program code and local variables are contained in local memory so that PE's emit requests only pertaining to global data references; and a system where all instructions and data are stored in global memory modules.

In the first case the SMP model may be applied "directly", i.e., the program "flowchart" may be used as the program which describes PE activity. Various SMP model parameters may be deduced by an "SMP model compiler" or by the SMP model user. SMP model statistical parameters may consist of procedure execution times if they do not entail references to data contained in global memory. In this case the complexity of the SMP model solution is dependent mainly on the number of states in the PE description diagram (i.e., the flowchart) and the number of PE's and GRM's, there is nearly a one-to-one mapping from program states in the high level flowchart to SMP model states. In this case *explicit* program execution time components may be computed.

In the second case where GRM references are very frequent, an instruction execution level of description may be used as the program flowchart, but its analysis may be practically intractable. (The SMP model's cost of solution becomes a problem when there are a significant number of states in the SMP state diagram -- systems of linear equations must be solved, their size is given by the number of states in PE programs.)

To alleviate such tractability problems, the modeler may consider state space reduction techniques or approximations. For example, source program procedures might be approximated by instruction models which are executed repeatedly. The number of instructions executed in a procedure must still be determined in some manner -- whether it be in the model or experimental domain is not important. Procedure models may then be connected together as the program control structure indicates.

The SMP model user is free to do appropriate model construction that may entail state lumping, etc. Our emphasis has not been to study such lumping procedures and state space reduction techniques. SMP model users are assumed to have done the necessary transformations to arrive at an SMP whose characteristics reflect the desired modeling results and is tractable. Including the program description of PE activity and GRM queues leads to the system depicted in figure 1.2. In the system, SMP's which describe

PE activity "drive" the resource system through their emissions of requests for service. Two types of states are shown: computation states and reference states (the latter are distinguished by a small dot). The dashed lines indicate the request emissions from reference states, they are dynamic links that exist for as long as connections are held. The sojourn time in reference states is a function of this connection time, which in turn is dependent on the queue activity at the GRM that receives the reference. By solving for the sojourn times of the SMP model, quantitative answers can be given to questions such as: how long it takes, on the average, to move from one SMP state to another; what fraction of time GRM's are busy; what fraction of time processors spend computing; how long GRM request queues are; and how long processors have to wait for their GRM connections.

1.2. Previous Models

Models of the system described have been developed in a simplified form. Multiprocessor models (which are based on a crossbar of negligible delay) have been developed in a simpler manner [BaS76, Bha75, Hoo77, MuM82a, Rau79, SeD79, SkA69, Smi74, Str70] where a simple GRM interface description of PE behavior has been assumed. PE's have been assumed to be in one of two states at any point in time: a computation state; and a GRM reference state. A procedure has been developed here for use in the SMP model solution that maps a given program to an equivalent two state process which exhibits similar low level behavior. This reduction technique is an approximation that provides an analytic approach to obtaining previous model parameters from the "source" programs.

These previous models have assumed that low level PE behavior statistics are available. Since, though, this data depends on the program under examination, their approach is a "post-programming" technique. That is, they depend on the availability of program object code statistics. If a model is to be used as a prediction mechanism, then such an approach is tedious at best. Since the SMP model is more closely associated with source program statistics, the approach developed here might be termed a "pre-programming" technique.

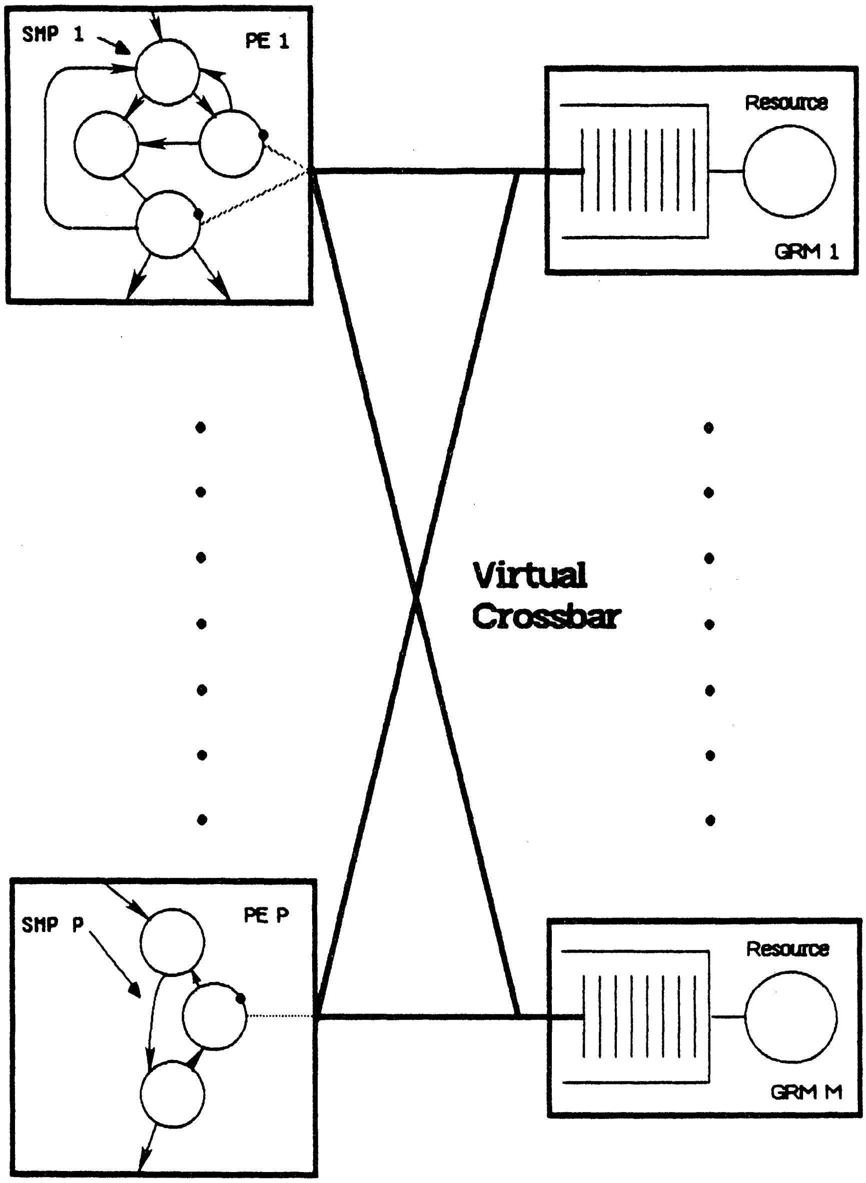


Figure 1.2 The system with SMP's and GRM queues.

Previous models' restriction to describing low level machine statistics such as GRM utilizations, queue lengths, and request queueing times limits their scope and utility. From such low level data little may be deduced regarding program execution times without extensive analysis after solving model equations. For example, loop and instruction execution times may not be predicted directly from the model solution.

Using a restricted set of performance measures may lead to inaccurate conclusions regarding programming choices. In fact, if resource utilizations are used as the only performance measure guide, incorrect choices regarding program execution times may be made altogether. The level and sophistication of performance measures is highly critical if a model is to be used as a an accurate evaluation tool.

Some effort has been made in analyzing multistage ICN's [MaG81, MuM82b, MuM82c, Pat79] but the results have not been derived in a form that is readily useful in the SMP model. They have typically assumed that PE's emit requests with a given mean rate. The CDF (cumulative distribution function) of the interemission time varies from one model to another. In these systems, a PE emits a request for service as above, requests may then be "shifted" through the ICN to their destination.

[Ram65] described the modeling of program execution on a uniprocessor as a Markov chain. This is a special case of the SMP model.

All the previous models of the system behavior referenced earlier are special cases of the SMP model although there are differences in the particular calculations used in, for example, computing request queueing times.

The SMP model may also be applied to non-computer applications where PE's are replaced by general logical *requestors*. Requestors need not be physical devices. For example requestors may correspond to customers which flow through a network of queues (resources). Hence the SMP model may also be applied to finite queueing network analysis where each network customer would correspond to a requestor whose behavior is equivalent to the customer's flow through the network (customers become requestors).

1.3. Organization

This thesis is organized as follows:

- Chapter 2 defines performance measures for the system described along with a discussion of classes of performance measures and performance measure importance.
- Chapter 3 describes the system model and its assumptions. Model parameters and system quantities are defined. Performance measure expressions are derived in terms of the model parameters and supplementary system quantities for most of

the performance measures defined in chapter 2.

- Chapter 4 completes the model calculations with the derivation of expressions for queueing times. The results of these calculations may be generally useful in small, finite customer queueing networks.
- Chapter 5 contains general system relationships that have been derived and discusses additional information and bounds on system behavior. Supplementary material concerning system relationships and properties are described here.
- Chapter 6 describes simulation studies that have been used to demonstrate and validate the model for both high level program execution experiments and low level synthetic experiments pertaining to calculation accuracy.
- Chapter 7 exemplifies the use of the model in a cache memory design study. It shows how the SMP model might be used as a comparison tool used in describing the impact of design choices on system performance.
- Chapter 8 describes model extensions and problems that may be considered in further work.
- Chapter 9 is the conclusion.

CHAPTER 2

PERFORMANCE MEASURE DEFINITIONS

The performance measures defined here may be partitioned into four basic classes: user class performance measures; PE class performance measures; GRM class performance measures; and ICN class performance measures. Each will be described next.

2.1. User Class Performance Measures

This class of measures is indicative of system performance as seen by system users, e.g., programmers. The measures reflect the performance of "high level" behavior such as execution times and response times to user input. Previous models have not been capable of predicting these quantities directly. The main quantities of interest here are:

- State transition times - these are the times required to traverse a given program segment. Note that users may perceive that transition times (of which program execution time is a specific case) are random variables because transition times depend on characteristics of input data and/or the sequence of conditional branches taken at runtime. This uncertainty in transition times reflects dependence on decisions made by programs as a result of input data values.

Notice that the majority of programs behave as finite state machines where transitions are either taken or not, there may be no random trajectory associated with typical programs *when given the initial state of the system.* (The exceptions are those programs that use random numbers to make decisions.) In other words, for a given initial state of the system, the program trajectory may be predicted in a deterministic manner. Due to uncertainty associated with input data and its effect on program execution trajectory, users may *perceive* execution time as a seemingly random quantity. (Operating system effects are ignored in this discussion.) Although typical programs only behave *seemingly* randomly, the model uses the approximation that they *actually* behave randomly. Note that contention for GRM's induces some *actual* randomness into a less than actually random program trajectory.

Since the assumption of actual randomness approximates seemingly random behavior, effort has been directed toward determining the appropriateness of this approximation. In retrospect, it has been found that this approximation works well in many circumstances, even where it was not expected to hold.

Consider, as an example of the use of transition times, the development of a simple system model of job execution. Figure 2.1 indicates some obvious possibilities. These models view servers (composed of a CPU, main memory, required disks, etc.) as servers for a job queue. The job queue might be modeled as an M/G/1 or M/G/P system. Paramount to using an M/G/P model is the determination of the first two moments of job execution time. The model developed here may be used to obtain the first two moments of job execution time.

State transition times in programs may be used to predict response times. For example, suppose a program checks a message queue upon entering any state in a set of states, then *set transition time moments* may be used to predict message response times and, for example, message queue lengths. Details will be defined and derived in chapters 3, 4, and 5.

Note that state cycle times (the time between entries to a given state) represent loop execution times in the case of recurrent states. Loop cycle times or their inverses represent *execution rates*.

An obvious application for execution time statistics is in program design and the physical organization of data structures in GRM's. If *no* data or instructions were stored in GRM's then there would be no degradation effects, on program p, due to resource contention. The model may in theory be used to optimize execution times.

Along with all performance measures defined here will be associated potential values denoted by a ^ over the corresponding quantity. These are computed by assuming no resource contention. (This may not always be a feasible operating point of the

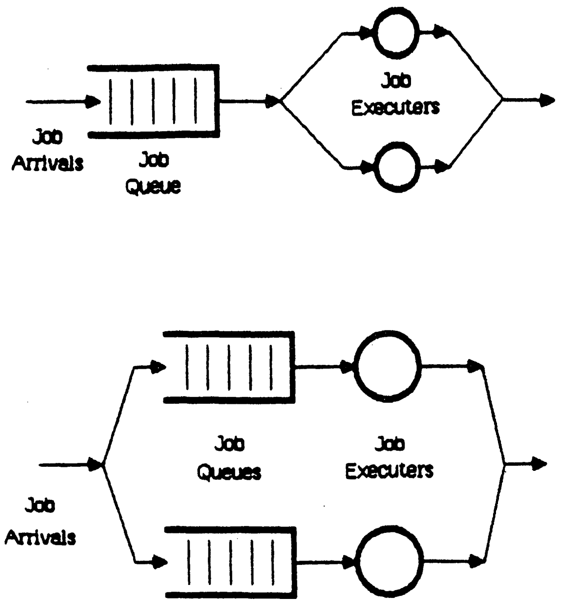


Figure 2.1 Simple system models.

system, chapter 5 discusses this.) They are useful in determining what might be achieved and in describing the behavior of a system which employs, in same manner, perfectly synchronized PE's. The potential values provide hard bounds on system behavior within the accuracy of the model assumptions. Potential values may also be used as objectives for comparison purposes. They provide a convenient measure of the amount of "processing power" lost due to resource contention.

2.2. PE Class Performance Measures

These performance measures are PE hardware performance measures. PE class performance measures are, for example, related to PE utilization and timing. The measures of interest in this class are:

- PE utilization - The fraction of time that a PE spends *not* referencing GRM's. As such, this measure indicates the "locality" of PE activity. If a PE's utilization is high then most PE time is spent in local computational activities, e.g., instruction execution or local memory accesses. Define this to be ϕ_p for PE p, $1 \leq p \leq P$.

- Potential PE utilization ($\hat{\phi}_p$) - This potential value indicates the inherent locality in PE p's program and its use of local hardware. It indicates the amount of PE utilization that can be achieved if there is no GRM contention.

- Relative PE utilization (ξ_p) - This is the PE utilization obtained when P programs are executing, relative to the potential value: $\xi_p = \phi_p / \hat{\phi}_p$. $1 - \xi_p$ indicates the fraction of "processing power" lost to GRM contention. ξ_p is more directly useful than ϕ_p or $\hat{\phi}_p$ in that it indicates PE utilization relative to its maximum. Note that ξ_p, ϕ_p, and $\hat{\phi}_p$ are not sufficient for comparing alternative programs which accomplish the same task. That is, two alternative programs may display opposite program execution times and ϕ_p's. One may have large program execution time and high processor utilization while another other may have small program execution time and a lower processor utilization. Notice that the *amount* of PE utilization lost to contention is $\hat{\phi}_p - \phi_p$.

- Coefficient of variation of PE request streams - The coefficient of variation of the time between PE request emissions indicates the amount of "randomness" in the request emission stream. Define this to be V_{P_j} for PE j. Note that this quantity exists formally if and only if the request intermission times are independent and identically distributed. This is certainly not true in most cases, it is though a typical approximation made in modeling [Kue79, MuM82b, MuM82c]. If $V_{P_j} = 0$, then the stream is highly deterministic; if $V_{P_j} = 1$ then the stream is "near" a Poisson emission stream; $V_{P_j} > 1$ indicates hyperexponential behavior. This quantity indicates the susceptibility of the system to "PE synchronization". The term PE synchronization will be used to describe the phenomenon which occurs when PE's alternate request emission because of request collisions during previous resource use. An example of this phenomenon will be seen in chapter 6. Intuitively it may be seen that there is higher susceptibility of the system to synchronize it all or several V_{P_j} are small (V_{P_j} represents the coefficient of variation with queueing included, a potential might also be a useful quantity here) than if several $V_{P_j} \simeq 1$, where there is sufficient randomness to ensure that synchronization will occur with negligible probability. The set of V_{P_j} informs the model user about the likelihood of PE synchronization. Although other parameters or quantities also determine the likelihood of PE synchronization (request rates for example) the utility of V_{P_j} should be clear.

2.3. GRM Class Performance Measures

GRM class performance measures concern the behavior of the system resources. The measures relevant here are:

- GRM utilization (ρ_m) - The fraction of time that GRM m is in use (over an appropriate interval defined by the model restrictions). This is predominantly the measure considered in previous models [Bha75, Hoo77, MuM82a, Pat79, Rau79, SeD79, Smi74, Str70].

- Outside observer's queue length ($\tilde{N}_m(t)$, at time t) - The number of requests in the GRM m queue at time t.

- Connection time ($\tilde{X}_m(n)$) - The connection time for the n^{th} connection between GRM m and any PE.

- Request arrival stream coefficient of variation (V_{M_m} for GRM m) - This is the coefficient of variation of the time between request arrivals at the GRM m queue. It indicates the amount of randomness apparent in the interarrival time and hence the waiting times experienced by arriving requests. Again the assumption that a renewal process exists at GRM queue inputs must be made, such is not the case in reality.

- Request rates of flow evident in the system in the steady-state.

GRM utilizations have typically been the performance measures predicted for multiprocessors of the sort considered. GRM utilizations, though, do not provide particularly useful information regarding PE and user class statistics. From ρ_m it is impossible to predict, for example, loop cycle times as described above. (Some "backwards" calculations may be used to find waiting times from ρ_m in special case models.) Some information may be obtained by using ρ_m in comparisons with other measures. For example, if ϕ_p's are low and ρ_m's are high it could be concluded that system operation is global resource service time bound. That is, GRM's are in use for a larger fraction of time than PE's are, hence to increase processing speed (decrease execution time) a design choice might be to increase GRM speed. The amount of increase would be reflected in the user class measures.

The pair ϕ_p, ρ_m (and possibly potential and relative values) may be used for intuitive comparison purposes relating to bottlenecks. The inclusion of the ICN in these measures might be done when it impacts system performance.

2.4. Virtual Crossbar Performance Measures

The quantities in this class are related directly to GRM activity but will be defined separately, they are:

- Bandwidth $(\tilde{B}(t))$ - the number of connections in existence between PE's and GRM's at time t. $\{\tilde{B}(t),\ t \geq 0\}$ is a fairly complicated stochastic process in general, as will be pointed out. Time limiting mean values will stressed. Time limiting pmf's might be derived for the number of connections in use.

- Rates of request flow - these will be introduced as required.

CHAPTER 3

MODEL DESCRIPTION

This chapter describes the SMP model framework. A PE state diagram is described, one is specified for each PE in the system. The PE state diagram (or stochastic finite state machine) consists of two types of states: computation states; and reference states.

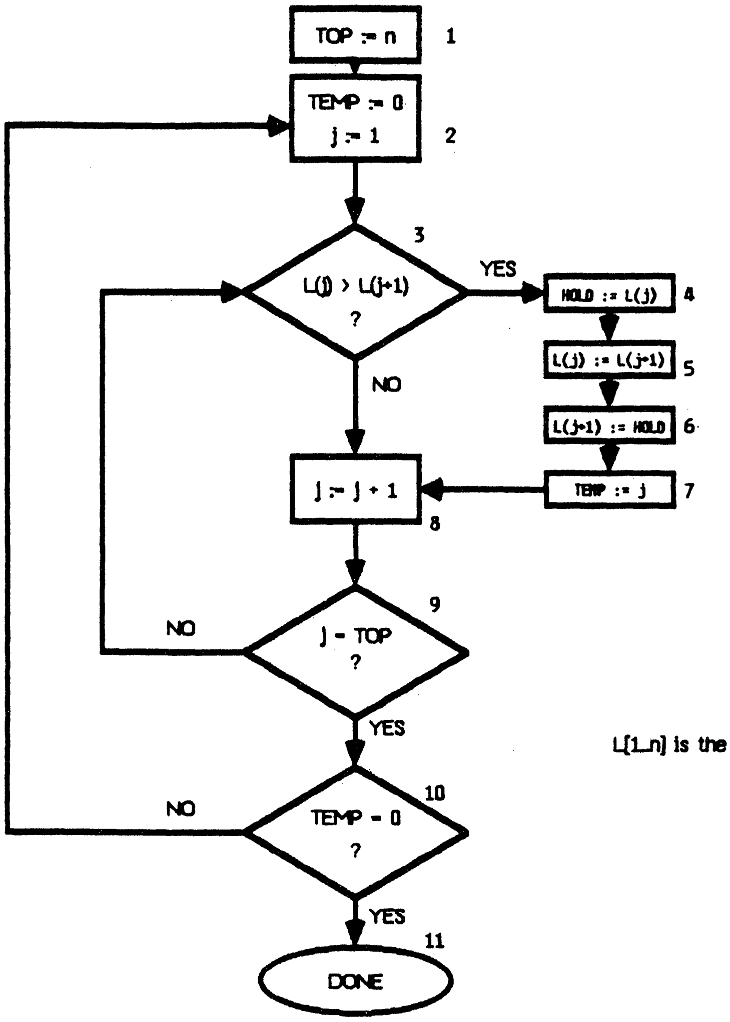
The set of states in the p^{th} PE state diagram will be referred to as A_p. The set of computation states in PE p's state diagram will be defined as Ω_{C_p}. Similarly, the set of reference states in PE p's state diagram will be termed Ω_{R_p}. States in PE state diagrams are either in Ω_{C_p} or Ω_{R_p} so $\Omega_{C_p} \bigcup \Omega_{R_p} = A_p$, $\Omega_{C_p} \bigcap \Omega_{R_p} = \phi$.

Ω_{C_p}, Ω_{R_p} and the graph joining their elements partially form the behavioral characteristics of PE behavior. Computation states are occupied while PE's execute internal operations, i.e., while they are not using global resources. Reference states are occupied while PE's reference GRM's. A PE may be waiting for service in a GRM queue or using a GRM during reference state occupation. The PE state diagram describes the transitions between computation and reference states.

The PE state diagram depends directly on system operation. It may in some circumstances represent program execution itself. Alternatively, it may represent low level PE behavior. That is, it may reflect elements of PE behavior using renewal based representations. Consider two examples to clarify the point.

First, consider an example of a high level description of PE behavior. In the exemplified system, processor instruction code and local data are stored in local memory (a system controller or operating system loads the program code into its assigned PE local memory) and GRM's store global data or large data structures. As an example of program execution on this system consider a bubble sort program (whose flowchart is shown in figure 3.1) where a list of length *n* is to be sorted. The list data is stored in GRM's due to the size constraints of local memory.

An appropriate PE state diagram that describes the bubble sort program is shown in figure 3.2. States in the original program that concern operations internal to PE's are computation states in the PE state diagram. Note that in general the mapping from a



L[1..n] is the list of data.

Figure 3.1 Simple bubble sort program.

source program to an appropriate PE state diagram is not unique. For example, figure 3.3 is also a valid PE state diagram. Source program states may be lumped to form a smaller PE state diagram. Conditional branches prevent a lumped PE state diagram from consisting of alternating computation and reference states.

Note that the time to move from state 1 to the final state 11 is the execution time of the program. The transition time in the corresponding PE state diagram is the execution time for the program.

PE's emit requests for GRM connections at their entries into reference states. That is, when a PE enters a reference state, it emits requests (for service) for the chosen GRM's. The formulation shown in chapter 4 assumes that one request is emitted when a PE enters a reference state. The reference state is said to terminate (in the single request per PE case) when the associated service has been completed. The GRM's chosen (i.e., the GRM's referenced) are determined by the resource system design and the layout of data in GRM's.[1] For example, if the GRM chosen is given by the lower $\lceil \log_2 M \rceil$ bits of the (physical) address generated by a processor (interleaved addressing) then the data list may be stored as shown in figure 3.4. Due to PE contention for GRM's, program execution time is influenced by data layout and GRM mapping hardware.

Consider next a system where instructions and all data are stored in GRM's. In this case a PE state diagram for the bubble sort program may be enormously complicated. Even in this simple program, the inclusion of instructions as computation states may make the PE state diagram enormous because it may be necessary to consider all individual instructions in the program. This fact (combined with the use of, for example, cache memory) may make it desirable to *represent* instruction execution with a *simple* representative model. The *level of description* chosen is determined partly by system operation and partly by the amount of detail desired.

Forms of approximating PE behavior using simple atoms executed repreatedly might be termed renewal based approximations in that they represent PE activity by reducing complicated behavior to a simpler approximately equivalent recurrent

[1] We will assume that only one request is emitted during GRM references. This assumption is not too restrictive in that problems occur if multiple requests per PE are permitted. Chapter 7 illustrates simple exceptions to this assumption while chapter 8 describes the problems involved in relieving it in general.

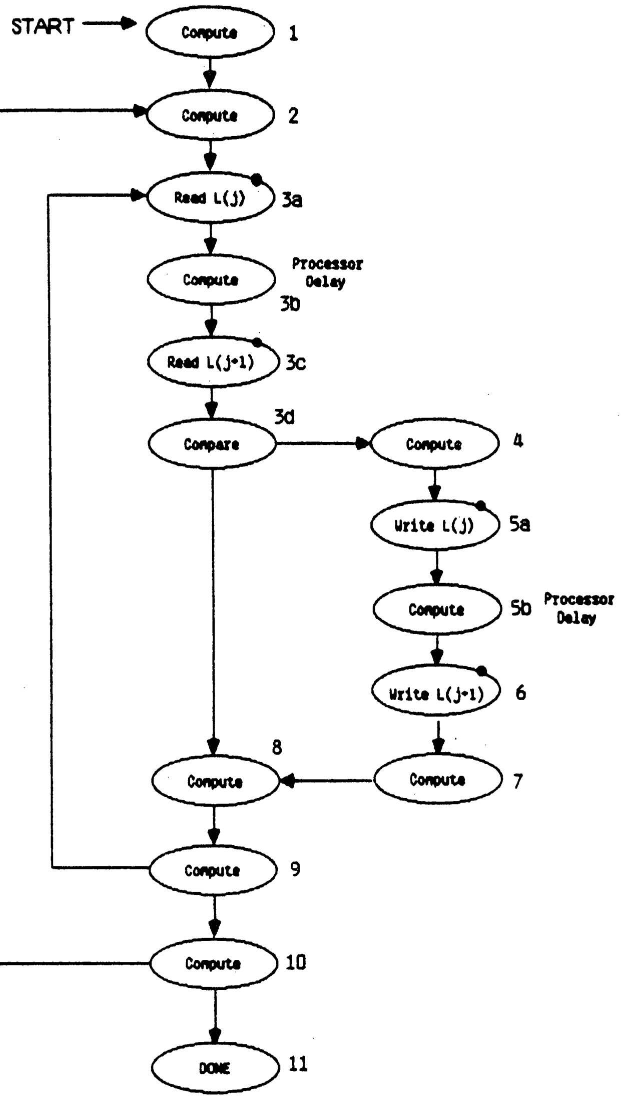


Figure 3.2 PE state diagram for the bubble sort program.

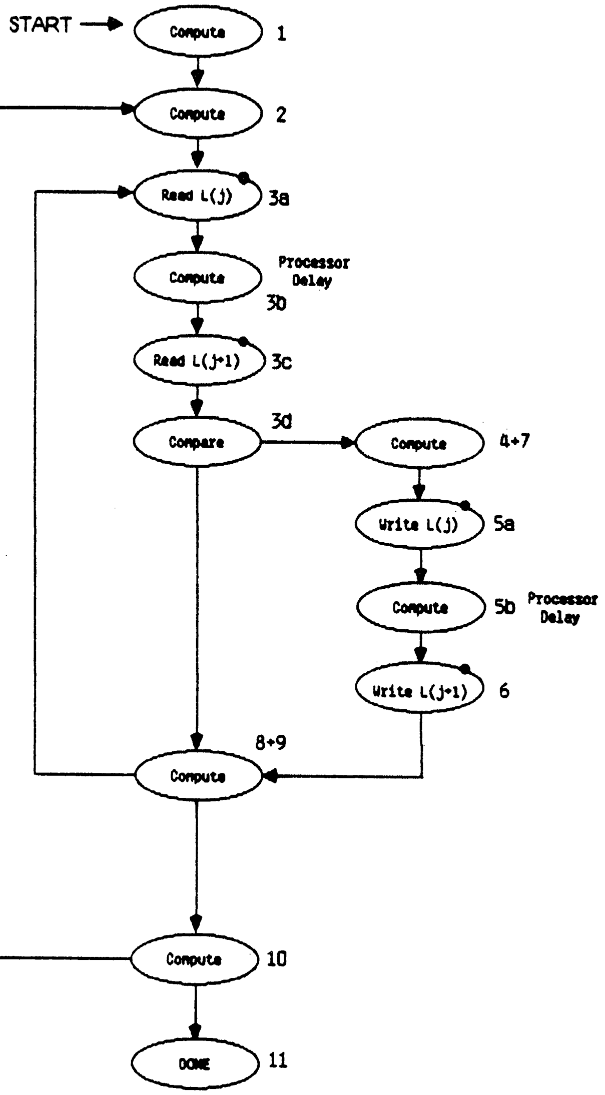


Figure 3.3 Alternate PE state diagram for the bubble sort program.

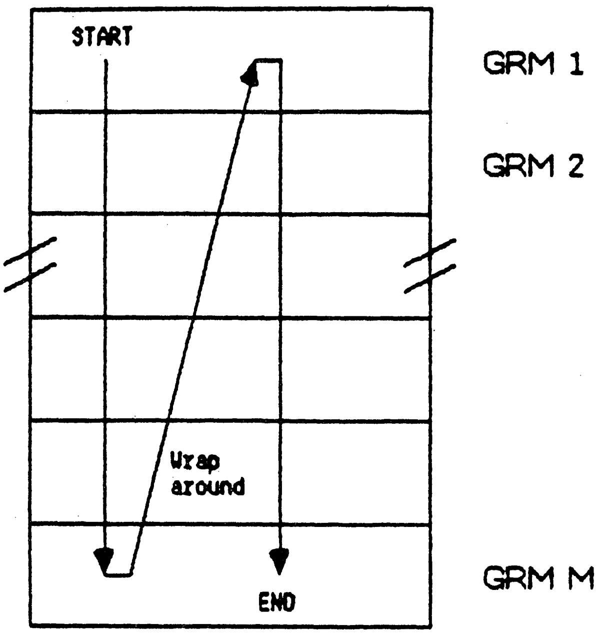


Figure 3.4 List layout under interleaved memory mapping.

description, the drawback is that user class measures might not be predictable directly using renewal based approximations.

For example, if rough estimates suffice and an instruction mix and hit ratios are known, then an appropriate instruction execution model is shown in figure 3.5 (this will be the subject of chapter 7). Program execution time may not be readily predicted from this PE activity model. State transition times now indicate instruction execution times, they do not indicate directly the source program execution time.

Also associated with the PE state diagram are the state transitions. That is, trajectories are associated with program execution. Transitions between PE states are taken to be instantaneous so that PE state diagrams behave as finite state machines.

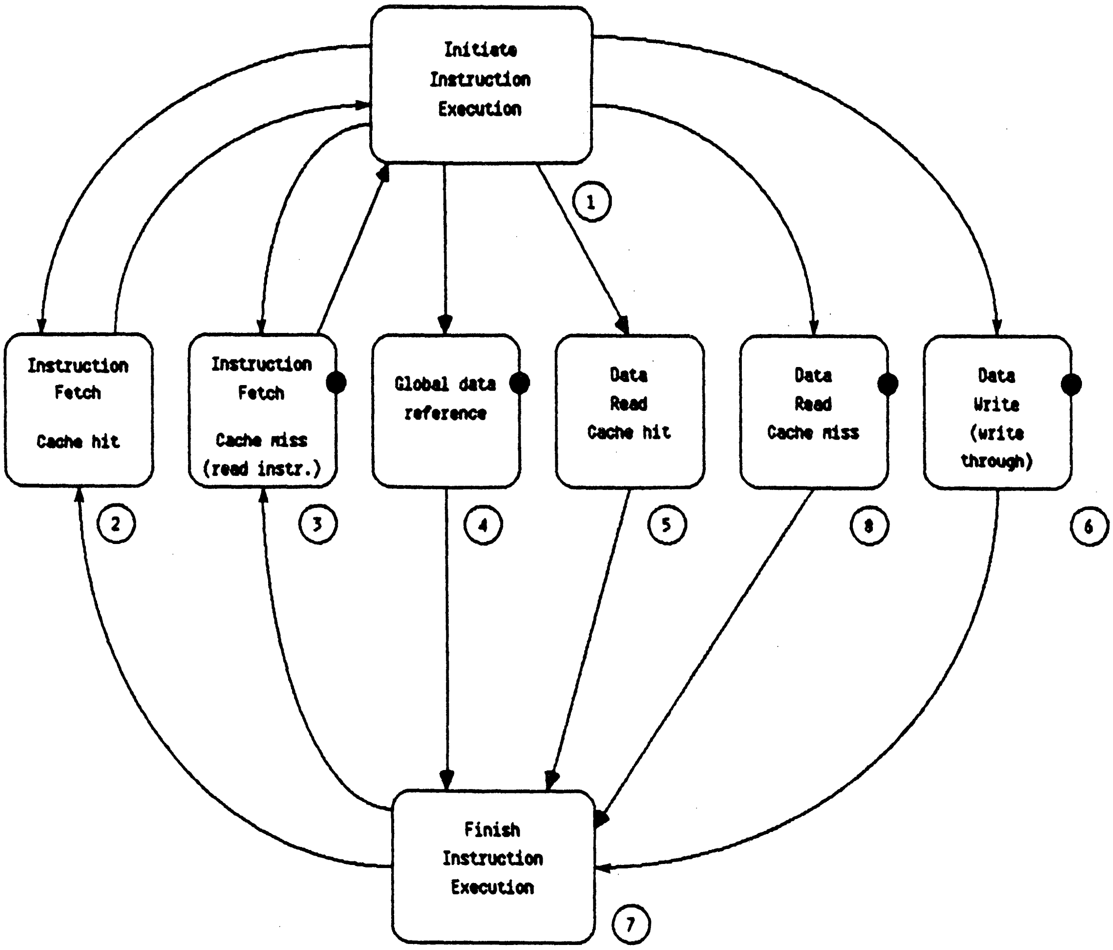

Initiate Instruction Execution
Instruction Fetch Cache hit
Instruction Fetch Cache miss (read instr.)
Global data reference
Data Read Cache hit
Data Read Cache miss
Data Write (write through)
Finish Instruction Execution
1
2
3
4
5
8
6
7

Figure 3.5 Instruction execution with cache memory.

3.1. The Model, Assumptions, and Preliminary Model Aspects

3.1.1. Overview

The following model description and analysis are organized in an effort to describe (compute) characteristics of PE state sojourn time CDF's as follows:

(1) Moments of PE state sojourn times will be emphasized.

(2) Moments of sojourn times may be written in terms of moments of GRM queueing times and GRM service times.

(3) Moments of GRM queueing times may be expressed (in several approximate ways) in terms of the sojourn times from 2. It is imperative that queueing time expressions obtained be independent of the expressions that relate sojourn times to queueing times as written in 2.

(4) After adequate expressions have been obtained that relate sojourn times to queueing times and queueing times to sojourn times, the solution aspect of the analysis may proceed. The solution aspect has not been emphasized, the derivation of accurate analytic expressions has been deemed more critical than accurate solution techniques -- there is little point to using an elegant solution technique on inaccurate expressions.

In practice, the simple solution scheme (a matrix based fixed point iteration scheme) has been found to be fast when compared to simulator execution time (less than 15 iterations are typically required in order for stable results to be obtained). Two waiting time formulations have been developed. It has been found that when one does not converge, the other converges acceptably.

3.1.2. The Model Assumptions

The model is based on the approximation that all P PE state machines (that correspond to PE state diagrams or programs) behave as semi-Markov processes (SMP's).

More formally, define $\tilde{Z}_p(t)$ to be the state (the state of PE p will taken to be an integer from 1 to $|A_p|$) of PE p at time t. *The SMP model is summarized by taking* $\{\tilde{Z}_p(t),\ t \geq 0\}$ *to be an SMP for all* p.

The approximation that $\{\tilde{Z}_p(t),\ t \geq 0\}$ is an SMP is equivalent to the following approximations regarding the sequencing of PE p's state machine:

At every PE p state transition, the joint distribution of the next state entered, and the time at which the present state is exited is dependent only on the state occupied at the present time. Formally, define the following notation:

$$\tilde{M}_p(n) = \begin{cases} \textit{the state of PE p after} \\ \textit{the } n^{th} \textit{ transition of its} \\ \textit{embedded Markov Chain (MC)} \end{cases}$$

By assuming $\{\tilde{Z}_p(t),\ t \geq 0\}$ to be an SMP, there exists an embedded MC. Let

$$\tilde{S}_{ps}(n) = \begin{cases} \textit{sojourn time of PE } p \\ \textit{in state } s \textit{ after the} \\ n^{th} \textit{ embedded MC} \\ \textit{transition} \end{cases}$$

Consider some ancillary information available from SMP properties. The approximation that $\{\tilde{Z}_p(t),\ t \geq 0\}$ is an SMP is equivalent to the following approximation:

$$\begin{aligned} &Pr(\tilde{S}_{ps}(n) \leq t,\ \tilde{M}_p(n+1) = j \mid \textit{all previous process (PE) p information}) \qquad (3.1) \\ &\simeq Pr(\tilde{S}_{ps}(n) \leq t,\ \tilde{M}_p(n+1) = j \mid \tilde{M}_p(\varsigma),\ \varsigma \leq n) \\ &= Pr(\tilde{S}_{ps}(n) \leq t,\ \tilde{M}_p(n+1) = j \mid \tilde{M}_p(n) = s) \end{aligned}$$

Note that conditions for approximate PE state machine independence may be written:

$$\begin{aligned} &Pr(\tilde{S}_{ps}(n) \leq t,\ \tilde{M}_p(n+1) = j \mid \textit{all previous system information}) \qquad (3.2) \\ &\simeq Pr(\tilde{S}_{ps}(n) \leq t,\ \tilde{M}_p(n+1) = j \mid \tilde{M}_p(n) = s) \end{aligned}$$

By the approximation that $\{\tilde{Z}_p(t),\ t \geq 0\}$ is an SMP, the dependence of $\tilde{S}_{pj}(n)$ on n may be dropped (state s entry epochs form a renewal process) to give a more concise form. Independent PE activity may also be written as:

$$\begin{aligned} &Pr(\tilde{Z}_1(t) = s_1,\ \tilde{Z}_2(t) = s_2,\ \ldots,\ \tilde{Z}_P(t) = s_P) \qquad (3.3\bullet) \\ &\simeq \prod_{p=1}^{P} Pr(\tilde{Z}_p(t) = s_p) \end{aligned}$$

Define the semi-Markov kernel for PE p to be

$$Q_p(i,j,t) = Pr(\tilde{S}_{pi} \leq t,\ \tilde{M}_p(n+1) = j \mid \tilde{M}_p(n) = i). \qquad (3.4)$$

If the transition probabilities $(Pr(\tilde{M}_p(n+1) = j | \tilde{M}_p(n) = i) \equiv P_p(i,j))$ are independent of sojourn times, then $Q_p(i,j,t) = S_{pi}(t)\, P_p(i,j)$. In this case, PE p enters state i, it stays there for $\tilde{S}_{pi}$ units of time and then moves to its next state. The next state is j with probability $P_p(i,j)$, independent of $\tilde{S}_{pi}$. $\mathbf{P}_p$ is the one step probability transition matrix for PE p's embedded MC. $S_{pi}(t)$ is, by definition, the CDF of PE p's sojourn time in state i. The above separability of the semi-Markov kernel seems to be a reasonable assumption for most programs (or PE state machines). This assumption *is not a* *strict requirement of the model* but equations have been written assuming this separability exists. Equations could certainly be rewritten to reflect non-separable cases. Without loss of generality then, we will use the separability to decompose the problem of studying $Q_p(i,j,t)$ into two parts: finding $P_p(i,j)$, and finding $S_{pi}(t)$.

In theory $Q_p(i,j,t)$ may be determined and from it cycle time and state first passage time CDF's may be determined. See [MaM82] for appropriate transform equations derived from [Cin75]. In practice it is too difficult to obtain $S_{pi}(t)$ so that $Q_p(i,j,t)$ cannot be reasonably calculated. At best the first few moments of sojourn times may be obtained and from them the first few moments of loop cycle and state first passage times may be approximated. Due to the fact that assuming PE's behave as SMP's is an approximation, it would not be worth developing more than the first few moments of timing measures.

3.2. Determining Transition Probabilities

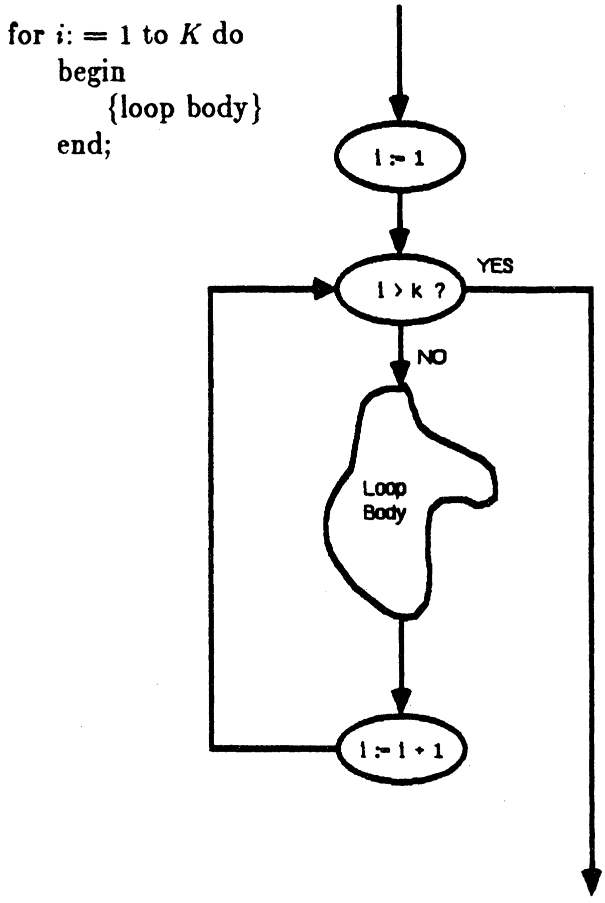
$P_p(i,j)$ represents the fraction of transitions that lead PE p from state i to state j. When $P_p(i,j)$ does not depend on time, it *is* a probability. As an approximation *transition fractions will be taken to be probabilities.* It is an approximation in that it ignores obvious correlation (time dependence) that may exist in, for example, 'for/DO' loop executions. This approximation has been used and tested during validation studies (described in chapter 6) with actual programs and has been found to work well.

The deduction of these relative frequencies may be done in many ways. If a program is well understood, then $P_p(i,j)$ may be derived from known results. For example, [Knu73] contains results on sorting algorithms that may be of use in determining transition probabilities.

An alternative to the analytic deduction of $P_p(i,j)$ is an experimental approach. Relative frequency data might be collected during uniprocessor program experimentation (since the fraction $P_p(i,j)$ is not queueing time dependent in non-real time applications, this is a "static" data collection process). This has been done traditionally when considering instruction mixes. There is certainly correlation between instruction sequences executed but simple instruction mixes have been used in successfully developing Huffman coding for op-codes.

If $P_p(i,j)$'s may not be determined, a reasonable approach to modeling is to use the unknown $P_p(i,j)$'s as parameters of the model. This would at least provide parameterized results.

Since loops occur so frequently in program execution, it is convenient to develop simple techniques for deducing $P_p(i,j)$ in these circumstances. First consider the modeling of simple 'for/DO' loops such as shown in figure 3.6. The branching probabilities are determined such that the ***first moment*** of the number of loops executed is equal to K. This first moment matching results in the distribution of the number of loops executed (in the model domain) being geometric. That is, loop execution in the model is



```
for i: = 1 to K do
    begin
        {loop body}
    end;
```

Figure 3.6 Loop modeling.

equivalent to a Bernoulli process (the occurrence of a "heads" in the flip of a coin corresponds to the exiting of the loop). When using first moment matching in deducing $P_p(i,j)$, higher moments of program execution times (which may be composed of many internal loops) may be inaccurately predicted. (For example, suppose $K = 100$ in figure 3.6, then E[number of loops executed] = 100 from first moment matching while the coefficient of variation of the loop completion time is 0.995, this would indicate characteristics "close" to an exponentially distributed loop execution time. Clearly, this is a false result. To alleviate the inaccuracy in higher moment predictions that results with first moment matching, an analysis using more detailed program specifications must be used; or the loop contents may be replicated K times in the PE state diagram.) The primary concern here will be with first moments although second moments (or at least approximations) will be required within the analysis.

When loop counts are determined by variables having associated pmf's at loop entry time, these pmf's may be used to find the mean number of loop executions and the above first moment matching may be done. If loop termination is caused by action within the loop, data might be collected experimently to be used in the model.

In low level PE state machines (where PE interaction with GRM's is described using a renewal based description of program execution) transition probabilities may indicate quantities such as instruction mix frequencies and cache memory hit ratios. These low level quantities are almost exclusively collected experimentally.

The model is also appropriate for system evaluation applications where a set of benchmark programs would be developed. That is, a set of program representations could be developed for system evaluation.

3.3. Determining Sojourn Times

The remaining properties of $Q_p(i,j,t)$ are contained in $S_{pi}(t)$. This section describes simple equations describing system operation and $S_{pi}(t)$. *The approach to model solution will be to write pertinent relationships describing the system and then use them in a solution scheme described in chapter 4. The main emphasis has been on equation development and model theory, not elegant solution techniques.*

Recall that there are two classes of states in the PE state machines: computation states (Ω_{C_p}); and reference states (Ω_{R_p}). When a PE is "in" a computation state it makes no GRM references (it is doing internal operations). Reference state timing is composed of two substates: a state in which PE requests are queued in their referenced GRM queues; and a state in which they use the desired connection. Then the sojourn time for PE p in state s when it references GRM m is:

$$\tilde{S}_{ps} = \tilde{W}_{psm} + \tilde{Y}_{psm} \qquad \textit{if GRM m is referenced} \tag{3.5•}$$

Where $\tilde{W}_{psm}$ is the queueing time experienced by a request from PE p when PE p enters state s and references GRM m. $\tilde{Y}_{psm}$ is the connection time of PE p using GRM m in state s. The CDF $Y_{psm}(t)$ maps the length of blocks transferred (in, for example, words) into a transfer (connection) time. This mapping reflects the speed and design of GRM m. It could represent GRM RAM speed, disk access time, etc. In general $Y_{psm}(t)$ describes the amount of service time desired by requestor p entering state s using GRM m. Note that *it is this relationship (3.5)* that would be modified to include ICN delays. An additive term could be included to represent ICN delays. Contention in ICN structures may require delay analysis.

Let the GRM chosen by PE p upon entry to state s be chosen according to the probability mass function η_{psm}. Then the first k moments (we are at most interested in the first few moments) of $\tilde{S}_{ps}$ are found from

$$\overline{S_{ps}^{k}} = \begin{cases} \sum_m E\left[(\tilde{W}_{psm} + \tilde{Y}_{psm})^k\right] \eta_{psm} & s \in \Omega_{R_p} \\ \textit{specified by the model user} & s \in \Omega_{C_p} \end{cases} \tag{3.6•}$$

Note that $\sum_m \eta_{psm} = 0$ for $s \in \Omega_{C_p}$, $\sum_m \eta_{psm} = 1$ for $s \in \Omega_{R_p}$ (with one request per PE allowed in the system). Multiple requests per PE might arise in many ways. In general multiple requests per PE have not been found to be particularly useful in a monoprogrammed environment except in certain cases discussed in chapter 8. Chapter 8 also discusses the problems involved with the general case. Equations will be written for the single request per PE situation.

If this relationship is re-written to display the particular reference involved (say the n^{th}) then:

$$\overline{S_{ps}^{k}}(n) = \sum_{m} E\left[(\tilde{W}_{psm}(n) + \tilde{Y}_{psm}(n))^{k}\right] \eta_{psm} \qquad s \in \Omega_{R_p}. \tag{3.7}$$

For every $n \geq 1$, $\tilde{W}_{psm}(n)$ and $\tilde{Y}_{psm}(n)$ are independent if there is no dependence of connection time (say the n^{th}) on the n^{th} request waiting time. It should be noted that $\tilde{W}_{psm}(n)$ is dependent on (at least) $\tilde{W}_{psm}(k)$ and $\tilde{Y}_{psm}(k)$ for $k < n$. That is, there is dependence on all other random variables in the system that took on values previous to the emission time of interest. The history dependent nature of the actual system makes the problem of accurate modeling difficult. Again, the use of SMP's in approximating PE behavior alleviates the history problem at the cost of some accuracy. It will, though, be important to include the *effects* from the history of processes in computations for finding $\overline{W_{psm}^{k}}$. For example, if $\lim_{n \to \infty} \overline{Y}_{psm}(n)$ is large, then a long queue (relative to a queue that would form if $\lim_{n \to \infty} \overline{Y}_{psm}(n)$ is small) may form behind the n^{th} PE p request as $n \to \infty$. This, in turn, might cause $\lim_{n \to \infty} \overline{W}_{psm}(n)$ to be large.

Throughout we will assume that $\tilde{W}_{psm}(n)$ converges in the first k moments, in the analysis in chapter 4 $k = 2$. This is not as strong a requirement as convergence in distribution. Convergence in moments here means the following:

$$\lim_{n \to \infty} E\left[\tilde{W}_{psm}^{k}(n)\right] = E\left[\lim_{n \to \infty} \tilde{W}_{psm}^{k}(n)\right] \equiv E\left[\tilde{W}_{psm}^{k}\right] \equiv \overline{W_{psm}^{k}} \tag{3.8}$$

exists (the bounded convergence theorem is used to move the limit inside the expected value). The history effects may be written as

$$\overline{W_{psm}^{k}}(n) = F\left(\overline{W_{psm}^{k}}(1), \overline{W_{psm}^{k}}(2), \cdots, \overline{W_{psm}^{k}}(n\text{-}1)\right) + H(\bullet) \tag{3.9}$$

Where $H(\bullet)$ represents functional dependence on other system quantities (not waiting times) not explicitly shown in $F(\bullet)$. F is simply a representative function displaying the dependence of $\overline{W_{psm}^{k}}(n)$ on previous values. Subsequently limit existence will be assumed for $\overline{W_{psm}^{k}}$.

To make analysis reasonable, time limiting (steady-state) values will be computed. Systems studied in simulation tests have been found to reach a steady-state (at least to within the desired modeling accuracy for those performance measures described above), typically within a few hundred GRM references. Occasionally attention will be given to limit existence in quantities when the analysis may yield new qualitative results.

The problem of determining the first few moments of $\tilde{S}_{ps}$ reduces to the problem of finding the first few moments of queueing times. This is covered in detail in chapter 4. In short, queueing and SMP analyses will be employed to complete the determination of $\overline{S_{ps}^{k}}$.

3.4. SMP Model Parameters

The SMP model parameters that arise (once any necessary reduction of an original program to its system dependent PE state machine has been done) may be determined in may ways. To summarize, the parameters and their determination are:

- $\mathbf{P}_p$ - the one step transition probability matrix for PE p. This is determined either experimentally or analytically depending on the complexity of the problem. First moment matching is done to ensure that means are computed appropriately. $P_p(i,j) = 1$ indicates a transition with certainty.

- η_{psm} - the probability that a request emitted by PE p when it enters state s is for GRM m. This parameter reflects the placement of data in GRM's and the type of function used to map physical addresses into GRM numbers. Note that the state dependence is convenient for handling special representation for particular GRM's. In examples shown later, some GRM's are taken to be disk units while others are taken to RAM (random access memory) modules. This allows certain states in PE state machines to act as disk I/O states while others represent RAM reference states. $\eta_{psm} = 1$ indicates an emission for GRM m with certainty.

- $Y_{psm}(t)$ - the connection time CDF for PE p using GRM m in state s. This CDF describes the characteristics of GRM m/PE p interactions. For example, if GRM m is a RAM module, then for simple read/writes the connection time CDF is $Y_{psm}(t) = u\,(t - \mathit{access\ time})$. For semaphore read/modify/write operations, $Y_{psm}(t)$ would indicate the amount of time required for an uninterrupted GRM connection. The manipulation of data structures that are stored in one GRM (for locking purposes) may be modeled in a unified manner by setting $Y_{psm}(t)$ appropriately. For disk modules, $Y_{psm}(t)$ might represent a seek time CDF. This parameter may be varied to study resulting performance changes in an effort to optimize system behavior and cost.

- $S_{ps}(t)$ - for $s \in \Omega_{C_p}$ this is the CDF of sojourn times for computation states. Hence this indicates the "speed" of PE local computations. Again, this may be varied to study the effects of changing PE speed on system performance. For $s \in \Omega_{R_p}$ moments of $\tilde{S}_{ps}$ are computed values.

3.5. Previous Models

Models that have been developed for memory interference [BaS76, Bha75, Hoo77, McC73, MuM82a, Pat79, Rau79, SeD79, SkA69, Smi74, Str70] are special cases of the SMP model in their representing processor activity as an alternating renewal process (an ARP, a two state SMP as shown in figure 3.7). They have done an accurate job of predicting ρ_m under special circumstances (described in the respective references). The utility of these models is limited to predicting GRM and PE class performance measures because of their assumption of low level PE descriptions. These models are based on low level system operation statistics such as the mean time between a GRM connection release and the next request emission. It will be seen that within the approximation that $W_{psm}(t) \simeq W_{pm}(t)$ (independent of the state), an SMP may be reduced to an ARP to obtain PE and GRM class performance measures.

[Ram65] considered the simple analysis (employing unsophisticated techniques that are more naturally handled using techniques described here) of a program modeled as a Markov chain, uniprocessor behavior only was considered. This corresponds to the model here with unit sojourn times and potential value results.

3.6. An Exact Approach

An exact approach to modeling system dynamics of the sort described here entails the study of an inherently transient coupled process such as $\{(\tilde{T}_{ps}(t))_{ps}\,;\, t \geq 0\}$ where $\tilde{T}_{ps}(t)$ is the amount of time spent by PE p in state s during $[0,t)$. The formulation of this approach would require the derivation of systems of probabilistic equations -- the approach would certainly be complex. It is believed that a transient analysis approach would prove to be more expensive (for the solution of the model equations) than program runs or simulations themselves. The formulation would also yield complicated

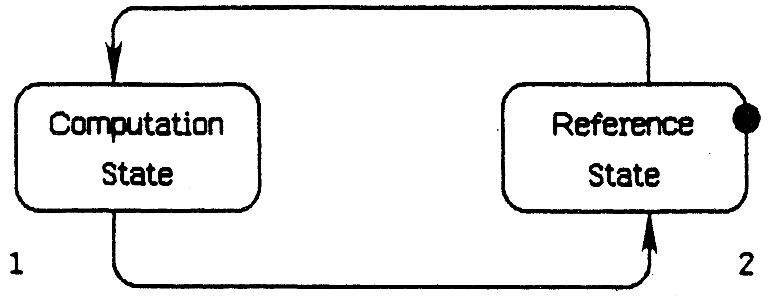


Figure 3.7 Simple ARP description of a PE.

time dependent CDF's $(T_{ps}(t,\alpha) = Pr(\tilde{T}_{ps}(t) \leq \alpha)$ for example) whereas the first few moments may suffice for the measures described in the SMP model. A purely analytic approach may yield such complicated relationships that nothing could be deduced from the results.

3.7. PE Independence

The model relationships that follow in chapter 4 are based on PE's behaving relatively independently. This approximation is most accurate in a general purpose, multiprogramming environment. The requirement need not be exactly satisfied, loose coupling will suffice as an approximation. Hence PE's (or more properly, the programs executing on PE's) which display infrequent communication or asynchronous communication with low message passing rates are the most applicable for the model. If PE state machines communicate through data structures infrequently, then it is reasonable to regard PE state machines as being statistically independent except through state occupation coupling caused by resource contention. Extensions of the model presented here that enable it to handle process communication using message queues (kept in global memory) are possible but have not been tested, see chapter 8.

3.8. Fundamental SMP Relationships

Elementary relationships and relevant facts regarding SMP behavior will be described, they may be found in common sources such as [Cin75, HeS82, Ros70]. Throughout the motivation will be that used in the context of describing PE p's SMP, so consider the stochastic process to be $\{\tilde{Z}_p(t),\ t \geq 0\}$.

Consider the *fraction* of time that $\{\tilde{Z}_p(t),\ t \geq 0\}$ spends in state s during an infinite interval. This fraction is only non-trivial for all states when the process is recurrent (non-null) for all $s \in A_p$. This in turn occurs for non-transient processes, that is, where any final (terminating) states in $\{\tilde{Z}_p(t),\ t \geq 0\}$ are connected to the PE's initial state, thereby simulating an infinite loop of program executions.

For example, figure 3.8 indicates the appropriate modification for an SMP with three terminal states. This "infinite loop" approach allows the model of the system to reach a a non-trivial steady-state.

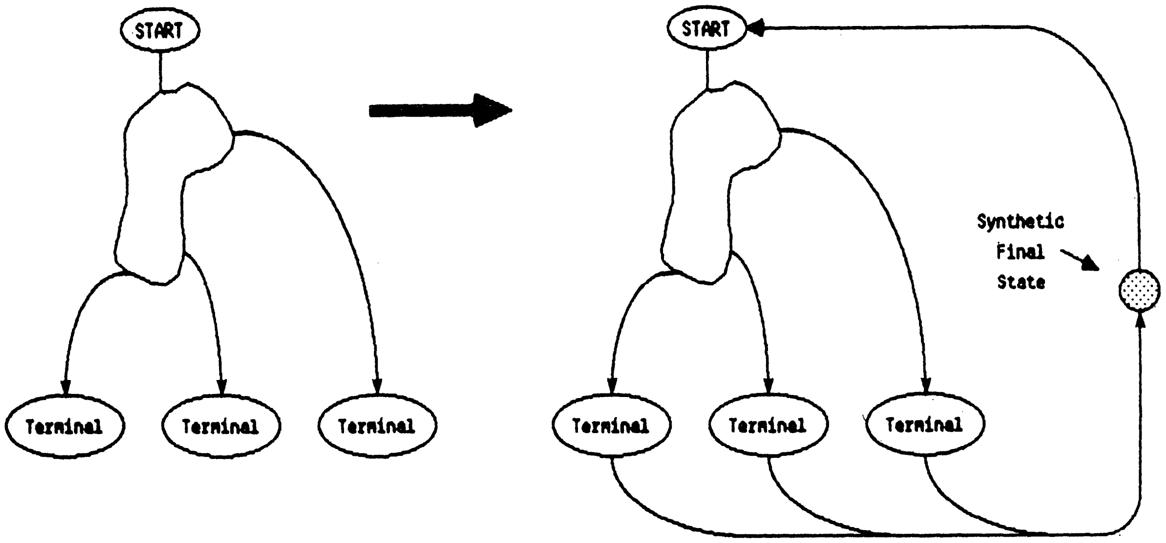


Figure 3.8 SMP loop construction.

Placing SMP's into an infinite loop makes transient and terminal states recurrent. This allows moments of execution time to be computed without calculating CDF's of transition times. That is, moments of transition times from an initial state to a final state may be computed by calculating the cycle time moments for the initial state after it has been connected to the final state. (A synthetic final state of negligible sojourn time might be imposed if necessary.) This ability derives from the fact that entries to state i form renewal epochs of a renewal process, hence steady-state results may be easily computed. Statistics may be computed regarding the renewal process created by the synthetic loop in a *simple manner*. Note that this is a much simpler solution process than the approach of computing the CDF of the transition times and then the moments.

A related point that must be addressed is the elimination of trivial initial states in PE programs, they contribute a known or negligible amount to execution times and may affect (adversely) resulting computations. Figure 3.9 shows an example. The state space resulting from the elimination of initial states will be termed the reduced state space (terminal states are connected to initial states in the reduced state space), call this A_p^r.

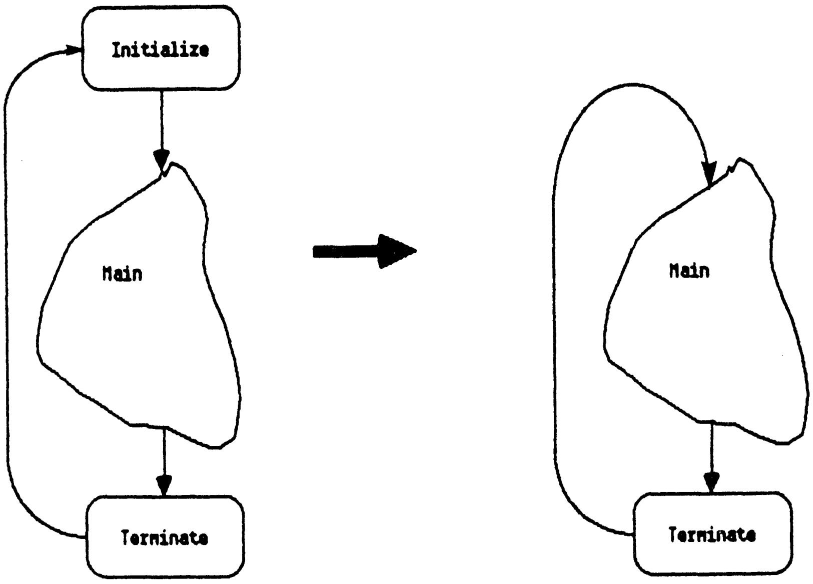


Figure 3.9 Initial state elimination.

Define p_{ps} to be the *fraction* of time that the reduced process $\{\tilde{Z}_p(t),\ t \geq 0\}$ spends in state s over an infinite interval. Consider $\{\tilde{Z}_p(t),\ t \geq 0\}$ to represent the reduced process from now on. Then from the approximation that $\{\tilde{Z}_p(t),\ t \geq 0\}$ is an SMP:

$$p_{ps} = \frac{\pi_{ps}\,\bar{S}_{ps}}{\sum\limits_{k \epsilon A_p'} \pi_{pk}\,\bar{S}_{pk}} = \lim_{t \to \infty} \frac{\int_0^t 1_{\{\tilde{Z}_p(\tau) = s\}} d\tau}{t} \tag{3.10}$$

Where π_{ps} is the embedded MC steady-state transition fraction (or probability if it exists) computed by solving $\boldsymbol{\pi}_p = \boldsymbol{\pi}_p \mathbf{P}_p$, $\sum\limits_{s \,\epsilon\, A_p'} \pi_{ps} = 1$. $\boldsymbol{\pi}_p$ is a row vector $(\pi_{p1}, \pi_{p2}, \cdots, \pi_{p|A_p'|})$. Note that p_{ps} appears to be first moment dependent on sojourn times. In the application here though the first moments of sojourn times $(\bar{S}_{ps})$ are dependent on higher moments of sojourn times. The dependence may be insignificant in some cases but may at times be quite pronounced. Hence, (3.10) is slightly deceptive in this respect. This relationship *does* help to explain though why system utilizations are *primarily* dependent on first moments of sojourn times as was experimentally noted in [Smi74]. The dependence on higher moments (although it may be slight) results from a detailed analysis of mean queueing times. In a general queueing situation (i.e., not a special case such as an M/G/1 approximation) the queueing time CDF is dependent on the CDF's of the arrival and service processes (the Lindley integral equation describes this in an idealized single GI/G/1 queue, [Kle75]) especially in a *closed system*. It is natural for the mean to depend on higher moments. This fact explains PE synchronization seen in simulation experiments. The dependence is an interesting (and unexpected) result of the queueing analyses in chapter 4. In most cases, the first moment of queueing times is the dominating factor in determining mean sojourn times, although the dependence on higher moments may occasionally be great.

It is interesting to study the conditions under which p_{ps} is a probability:

$$p_{ps} = \lim_{t \to \infty} Pr(\tilde{Z}_p(t) = s) \tag{3.11}$$

p_{ps} as described in (3.10) is always a fraction, it is also a probability as in (3.11) in several cases: when the cycle time[2], defined as $\tilde{C}_{pss}$, for successive entries to state s has a density on an interval (see [Cin75, HeS82, Ros70]) or we assume that $\{\tilde{Z}_p(t), t \geq 0\}$ is ergodic. p_{ps} is also a probability in another simple case:

Since the system under consideration is based on a discrete clock (at some level, there may be different clocks for different PE's, this is unimportant) these cycle times are certainly not absolutely continuous (they do not display a density on an interval). The transition points (i.e., jump points, or discontinuities) of $C_{pss}(t)$ determine in what sense p_{ps} is a probability. If $\tilde{C}_{pss}$ has lattice distribution (that is, transitions in $C_{pss}(t)$ occur on integral multiples (> 1) of a fundamental time period termed the "span" or "period" of the process) then p_{ps} from (3.10) is a probability as in (3.11) at the time limiting lattice points $\lim Pr(\tilde{Z}_p(n) = s), n \rightarrow \infty, n \in \{ \textit{ lattice points for } \tilde{C}_{pss} \}$. If $\tilde{C}_{pss}$ is discrete but of unclassified nature, the p_{ps} as in (3.10) is a probability at the transition points of $C_{pss}(t)$. For times between transition points (and a general discussion) of limits on $Pr(\tilde{Z}_p(t) = s)$, see [Cin75].

The limits have ramifications in approximations used later. For example, since PE's described here are based on a clock, the clock period determines the times at which PE state occupation fractions may be used to represent probabilities. This is important in that if a PE is viewed (by another PE, or an outside observer for example) at lattice points, then (3.10) may be used as a probability, but if the PE is examined at non-lattice points, then the use of (3.10) as a probability is an approximation. Since outside observers "see" averages, (3.10) may be used to predict outside observer statistics, conversely, since PE events may not occur at these special "Poisson" times, a PE viewing another PE at non-lattice time may not "see" the fractions as probabilities.

A useful relationship that is readily available is:

$$p_{ps} = \frac{\overline{S}_{ps}}{\overline{C}_{pss}} = \frac{\pi_{ps}\,\overline{S}_{ps}}{\sum\limits_{k \in A_p'} \pi_{pk}\,\overline{S}_{pk}} \qquad (3.12\bullet)$$

This provides a relationship between *mean* loop times, sojourn times, and p_{ps}. The approach to solving model relationships (to obtain a prediction) will be to determine $\overline{S}_{ps}$ indirectly as a function of p_{ps}. Then (3.12) may be used to determine p_{ps} as a function

[2] The terms loop time, recurrence time, and cycle time are used synonymously here.

of $\overline{S}_{pk}$. Note that (3.12) provides a straightforward relationship for determining mean state loop times -- one of the performance measures previously defined.

Important quantities in SMP analysis are the transition times for states to sets (of states) and from sets to sets. These are important in predicting moments of response times and mapping a given PE state machine to a "nearly" equivalent two state ARP activity model.

Define $\tilde{T}_p(i,\Omega)$ to be the time between *leaving* state $i \in A_p'$ and *entering any* state in $\Omega \subset A_p'$. To compute moments of $\tilde{T}_p(i,\Omega)$ for given i and Ω, in a similar manner to that used in [HeS82], condition on the state entered upon leaving state i. This approach gives:

$$\begin{aligned} \overline{T_p^k}(i,\Omega) &= \sum_{j\epsilon\Omega} 0\times P_p(i,j) + \sum_{j\epsilon A_p'-\Omega} E\left[\{\tilde{S}_{pj} + \tilde{T}_p(j,\Omega)\}^k\right]\times P_p(i,j) \\ &= \sum_{j\epsilon A_p'-\Omega} E\left[\{\tilde{S}_{pj} + \tilde{T}_p(j,\Omega)\}^k\right] P_p(i,j) \end{aligned} \qquad (3.13\bullet)$$

Which forms a relationship between $\overline{T_p^k}(i,\Omega)$ for all $i \in A_p'$. Note that because $\tilde{T}_p(j,\Omega)$ is the time required to traverse an SMP from state j to a state in Ω after leaving state j (see figure 3.10), $\tilde{T}_p(j,\Omega)$ is independent of $\tilde{S}_{pj}$. That is, given $\tilde{S}_{pj} = \tau$, $\tilde{T}_p(j,\Omega)$ is not influenced by $\tilde{S}_{pj}$ due to the sequential nature of the SMP transitions.

As many moments as required (say k) are computed in a recurrent manner, moments 1,...,k-1 are used to find the k^{th} moment. To compute $\overline{T}_p(i,\Omega)$ for a given Ω, equation (3.13) becomes a vector-matrix equation $\mathbf{A\overline{T}}_p(\Omega) = \mathbf{f}$.

From $\overline{T_p^k}(i,\Omega)$ the moments of transition times from a set to a set are computed directly. Denote by $\overline{T_p^k}(\Omega_1 \rightarrow \Omega_2)$ the k^{th} moment of the time between leaving set $\Omega_1 \subset A_p'$ (any state in Ω_1) and entering set $\Omega_2 \subset A_p'$. These moments are computed by conditioning on the state $i \in \Omega_1$ that was left when set Ω_1 was exited. Then unconditioning gives:

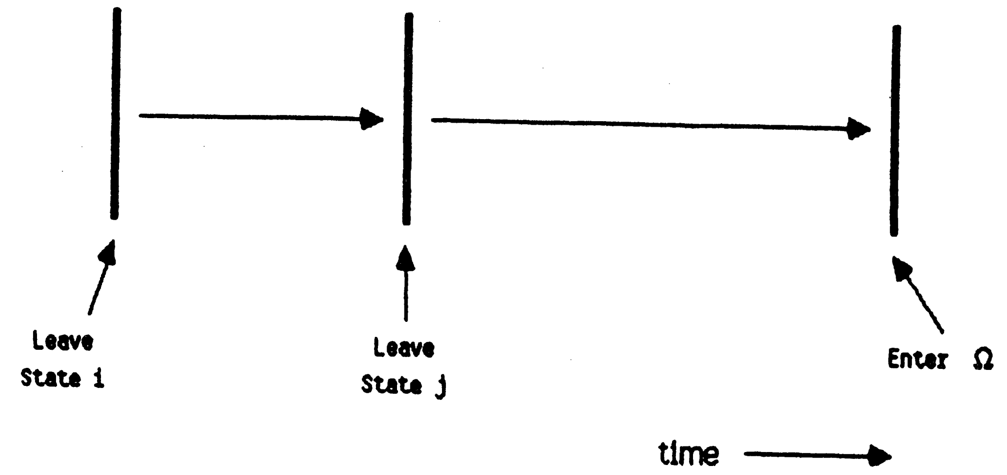


Figure 3.10 Set transition time independence.

$$\begin{aligned}
\overline{T_p^k}(\Omega_1 \to \Omega_2) &= \sum_{i\in\Omega_1} \overline{T_p^k}(i,\Omega_2) \left[\frac{\pi_{pi}}{\sum_{j\in\Omega_1}\pi_{pj}}\right] \\
&= \sum_{i\in\Omega_1} \overline{T_p^k}(i,\Omega_2)\,\frac{\pi_{pi}}{\pi_{p\Omega_1}} \qquad (3.14\bullet) \\
&= \frac{1}{\pi_{p\Omega_1}} \sum_{i\in\Omega_1} \overline{T_p^k}(i,\Omega_2)\pi_{pi}
\end{aligned}$$

These calculations may provide further information regarding process execution timing, for example if $\Omega = \{i\}$ then $\overline{T_p^k}(\Omega \to \Omega) = \overline{C}_{pii} - \overline{S}_{pi}$. If Ω is a set of states that represents an appropriate action (for example, suppose a program checks its message queue upon entering any of a set of states, then set transition times are required to compute moments of the time between queue examinations), then these transition times may represent moments of response times.

Aside from the utility of $\overline{T_p^k}(i,\Omega)$ in predicting set transition times is its ability to reduce a given SMP to an ARP with approximately equivalent GRM and PE class behavior. The temporal averages are maintained in the equivalence, although the instantaneous behavior of the two processes are different. This reduction relies on finding the first k moments of an equivalent computation time state (state 1 in figure 3.7). Here k represents the maximum number of moments that are deemed required within the model approximation accuracy. For example, waiting time calculations described later require the first two moments of computation state sojourn times. Previously developed models of the ARP process description require the first moment of the ARP computation state sojourn time. To find the first k moments of the equivalent computation state sojourn time, set $\Omega_1 \equiv \Omega_2 \equiv \Omega_{R_p}$ then $\overline{S_{p1}^{k}}' = \overline{T_p^k}(\Omega_{R_p} \rightarrow \Omega_{R_p})$. This provides a means for computing ARP description quantities from the original SMP. Note that because Bernoulli loops (used to represent iteration loops) only match the first moment of loop counts, higher moments may be inaccurately predicted unless the actual program is close to an SMP. That is, because first moment matching is done in finding the transition probabilities, error may occur when attempting to compute higher moment values.

3.9. Measure Derivation in the SMP Model

Expressions for many of the SMP model performance measures are described next, those not covered are expressed in chapters 4 and 5.

3.9.1. PE and System Utilizations

Consider PE p's utilization, it is given by:

$$\phi_p = \sum_{s \epsilon \Omega_{C_p}} p_{ps} \quad \mathit{and} \quad \hat{\phi}_p = \sum_{s \epsilon \Omega_{C_p}} \hat{p}_{ps} \tag{3.15•}$$

Where $\hat{p}_{ps}$ is the potential value corresponding to p_{ps}:

$$\hat{p}_{ps} = \frac{\pi_{ps} \hat{\bar{S}}_{ps}}{\sum_{k \epsilon A_p^r} \pi_{pk} \hat{\bar{S}}_{pk}} \tag{3.16}$$

and

$$\hat{\bar{S}}_{ps} = \begin{cases} \bar{S}_{ps} & s \, \epsilon \, \Omega_{C_p} \\ \sum_m \bar{Y}_{psm} \, \eta_{psm} & s \, \epsilon \, \Omega_{R_p} \end{cases} \tag{3.17}$$

From this it *appears* that ϕ_p is a first moment property, but again it must be noted that $\bar{S}_{ps}$ is actually a function of higher moments. The dependency is very complex and approximations (only) will be shown later in chapter 4. $\hat{\phi}_p$ is a first moment dependent quantity.

Again, relative PE utilization is computed directly:

$$\xi_p = \frac{\phi_p}{\hat{\phi}_p} \tag{3.18}$$

Note that a uniprocessor achieves its potential so $\xi_1 = 1$ for a uniprocessor. A system utilization might be defined:

$$\phi = f(\phi_1, \phi_2, \ldots, \phi_P) \tag{3.19}$$

$$\hat{\phi} = f(\hat{\phi}_1, \hat{\phi}_2, \ldots, \hat{\phi}_P)$$

$$\xi = \frac{\phi}{\hat{\phi}} \tag{3.20}$$

These definitions represent relative importance of various PE tasks. For example, a few PE's tasks may be designated as important relative to other tasks. ϕ could be defined to be most influenced by the important PE's so that ϕ measures system utilization with high priority tasks appropriately affecting system utilization. ξ could then be used as a comparison measure of system utilization. This particular measure has not been extensively used.

3.9.2. GRM Utilizations

Since these quantities are not directly available from SMP results, they must be derived from system behavior. First define the following rates which are useful in themselves:

$$\lambda_{psm} = \textit{rate of request flow from PE p to GRM m due to state s} \tag{3.21}$$

$$\lambda_{pm} = \textit{rate of request flow from PE p to GRM m}$$

$$\lambda_m^{i} = \textit{rate of request flow into GRM m}$$

$$\lambda_p^o = \textit{rate of request flow out of PE } p$$

The following relationships exist among such data.

$$\begin{aligned}
\lambda_{pm} &= \sum_{s \in A_p^r} \lambda_{psm} \\
\lambda_m^i &= \sum_p \lambda_{pm} \\
\lambda_p^o &= \sum_m \sum_{s \in A_p^r} \lambda_{psm} = \sum_m \lambda_{pm}
\end{aligned} \tag{3.22•}$$

Apply the theory of rewards in SMP's [Ros70] to compute λ_{psm} as follows:

$$\textit{define} \;\; r_s(\tilde{Z}_p(t)) = \begin{cases} \dfrac{\eta_{psm}}{\overline{S}_{ps}} & \tilde{Z}_p(t) = s \\ 0 & \textit{otherwise} \end{cases} \tag{3.23}$$

$r_j(j)$ indicates an emission (reward) rate of $\frac{\eta_{pjm}}{\overline{S}_{pj}}$ for GRM m during the time that process p is in state j (η_{pjm} requests are emitted for GRM m in an average of $\overline{S}_{pj}$ time units, hence the ratio is a mean rate of emission). Then

$$\begin{aligned}
\lambda_{psm} &= \lim_{t \to \infty} \frac{\int_0^t r_s(\tilde{Z}_p(\tau))d\tau}{t} \\
&= \sum_{k \in A_p^r} r_s(k) p_{pk} \\
&= \frac{\eta_{psm}}{\overline{S}_{ps}} \, p_{ps}
\end{aligned} \tag{3.24•}$$

This gives a simple and intuitive calculation for the process p emission rates. Note that $\lambda_{psm} = \frac{\eta_{psm}}{\overline{S}_{ps}} \, p_{ps} = \frac{\eta_{psm}}{\overline{C}_{pss}}$ from the fact that $p_{ps} = \frac{\overline{S}_{ps}}{\overline{C}_{pss}}$

Define $\tilde{X}_m(n)$ to be the connection (service) time of the n^{th} connection between a PE (any PE) and GRM m. Then let $\lim_{n \to \infty} \overline{X_m^k}(n) = \overline{X_m^k}$ be the k^{th} moment of the steady-state connection time for GRM m. Assume that the limit exists. $\overline{X_m^k}$ is computed as (see figure 3.11 for a graphical representation):

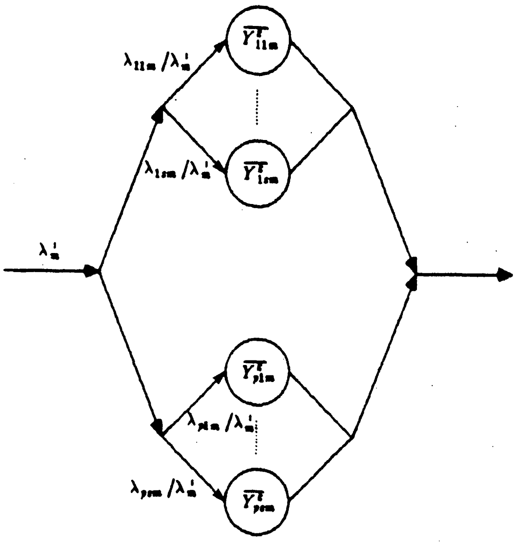

$\lambda_{11m}/\lambda^i_m$
Y^k_{11m}
$\lambda_{1sm}/\lambda^i_m$
Y^k_{1sm}
λ^i_m
Y^k_{p1m}
$\lambda_{p1m}/\lambda^i_m$
$\lambda_{psm}/\lambda^i_m$
Y^k_{psm}

Figure 3.11 Graphical representation of a GRM server.

$$\begin{aligned}\overline{X^k_m} &= \sum_p\sum_s \frac{\lambda_{psm}}{\lambda^i_m}\,\overline{Y^k_{psm}} \\ &= \frac{1}{\lambda^i_m}\sum_p\sum_s \lambda_{psm}\,\overline{Y^k_{psm}}\end{aligned} \qquad (3.25\bullet)$$

Notice that *fractions* ($\lambda_{psm}/\lambda^i_m$) are used here to approximate probabilities. This is inaccurate when the limit does not exist: consider that case where arrivals are not independent. For example, if PE request arrivals alternate -- one arrives from PE i then one from PE j, and another from PE i, etc. -- and mean connections are not the same for all requests, then no limiting mean exists for the connection time. The calculation (3.25) gives a request average mean: $\overline{X^k_m} = \lim_{n\to\infty} \frac{\sum_{j=1}^{n} \tilde{X}^k_m(j)}{n}$.

The use of fractions as probabilities is correct however if we are interested in the moments for an *arbitrary* time limiting outside observer's examination of the connection time random variable. That is, it provides the connection length random variable statistics at a time limiting Poisson examination time (not to be confused with the *excess* connection time that may exist for the time between the examination and the subsequent release of the connection).

There are at least three ways of calculating GRM utilizations, ρ_m. The first and simplest comes from queueing theory [GrH74, Kle75]:

$$\rho_m \;=\; \lambda'_m \bar{X}_m \;= \sum_p \sum_{s \epsilon A_p^r} \lambda_{psm} \bar{Y}_{psm} \;\equiv\; \sum_p \rho_{pm} \qquad (3.26\bullet)$$

Where ρ_{pm} is the "work load" (utilization) brought to GRM m by PE p. In particular $\rho_{pm} \;=\; \sum_{s \epsilon A_p^r} \lambda_{psm} \bar{Y}_{psm}$. This is the formulation used later in chapter 6 examples.

The next approach is based on viewing the GRM m server (controller) at a random point in time (t) as $t \rightarrow \infty$. (That is, let the system be in its time limiting steady-state at t.) Define a PE to be ***using*** GRM m if it has a request in queue m or is using GRM m. Then:

$$\rho_m \;= Pr(\mathit{GRM\ m\ is\ busy\ at\ t}) = 1 - Pr(\mathit{GRM\ m\ is\ idle\ at\ t}) \qquad (3.27)$$

$$\begin{aligned}
&Pr(\mathit{GRM\ m\ is\ idle\ at\ t}) = Pr(\mathit{all\ PE's\ are\ not\ using\ GRM\ m\ at\ t}) \\
&= \; Pr(\mathit{PE\ 1\ is\ not\ using\ GRM\ m\ at\ t,} \\
&\qquad \mathit{PE\ 2\ is\ not\ using\ GRM\ m\ at\ t,} \\
&\qquad \ldots \\
&\qquad \mathit{PE\ P\ is\ not\ using\ GRM\ m\ at\ t}) \\
&\qquad = Pr(\mathit{PE\ 1\ is\ not\ using\ GRM\ m\ at\ t} \mid \mathit{PE's\ 2\text{-}P\ are\ not}) \\
&\qquad \times \; Pr(\mathit{PE\ 2\ is\ not\ using\ GRM\ m\ at\ t} \mid \mathit{PE's\ 3\text{-}P\ are\ not}) \\
&\qquad \times \; Pr(\mathit{PE\ 3\ is\ not\ using\ GRM\ m\ at\ t} \mid \mathit{PE's\ 4\text{-}P\ are\ not}) \\
&\qquad \ldots \\
&\qquad \times \; Pr(\mathit{PE\ P\text{-}1\ is\ not\ using\ GRM\ m\ at\ t} \mid \mathit{PE\ P\ is\ not}) \\
&\qquad \times \; Pr(\mathit{PE\ P\ is\ not\ using\ GRM\ m\ at\ t})
\end{aligned}$$

If PE's are taken to be approximately independent, then a simple formulation may be written:

$$Pr(\mathit{GRM\ m\ is\ idle\ at\ t}) \;=\; \prod_p Pr(\mathit{PE\ p\ is\ not\ using\ GRM\ m\ at\ t}) \qquad (3.28)$$

Then:

$$Pr(\mathit{PE\ p\ is\ not\ using\ GRM\ m\ at\ t}) \;=\; \sum_{s \,\epsilon\, A_p^r} (1 - \eta_{psm}) p_{ps}$$

Recognize that dependence between PE state occupations at time t (because of GRM contention effects) makes this an approximation. An alternative technique might be to compute these conditional probabilities (or fractions) in an effort to increase accuracy. That is, to compute state occupation coupling between processes. The queueing approach has been used due to its simplicity.

The last obvious calculation for GRM utilizations is simply a variant on the approach above at time t. Here term "using" to mean accessing the GRM resource, not including time waiting for service.

This is based on viewing GRM m at time t but using a sum formulation.

$$\begin{array}{rl} \rho_m & = Pr(GRM\ m\ is\ busy\ at\ t \\ & \quad or\ PE\ 2\ is\ using\ GRM\ m\ at\ t, \\ & \quad \ldots \\ & \quad or\ PE\ P\ is\ using\ GRM\ m\ at\ t) \end{array} \tag{3.29}$$

Since use of GRM's is disjoint (e.g., not two PE's may simultaneously use a GRM), then the "or" condition may be evaluated using a simple sum:

$$Pr(GRM\ m\ is\ busy\ at\ t) = \sum_p Pr(PE\ p\ is\ using\ GRM\ m\ at\ t) \tag{3.30}$$

Notice that each reference state in a PE SMP may be viewed as the successive occupation of two substates: one in which the PE request is in the GRM queue; and one in which the PE uses the GRM. Then if t is a random (Poisson) time an appropriate formulation is:

$$\begin{array}{l} Pr(PE\ p\ is\ using\ GRM\ m\ at\ t \mid PE\ is\ in\ state\ s\ at\ t) \\ \quad = \eta_{psm} \dfrac{\overline{Y}_{psm}}{\overline{Y}_{psm} + \overline{W}_{psm}} \end{array} \tag{3.31}$$

Then unconditioning with the general time probabilities for the probability that PE p is in state s at time t gives (again we know that the SMP's are not independent so that this is also an approximation):

$$Pr(PE\ p\ is\ using\ GRM\ m\ at\ t) = \sum_{s \in \Omega_{R_p}} \eta_{psm} \left(\frac{\overline{Y}_{psm}}{\overline{Y}_{psm} + \overline{W}_{psm}} \right) p_{ps} \tag{3.32}$$

Therefore,

$$\rho_m \simeq \sum_p \sum_{s \in \Omega_{R_p}} \eta_{psm} \left(\frac{\overline{Y}_{psm}}{\overline{Y}_{psm} + \overline{W}_{psm}} \right) p_{ps} \tag{3.33}$$

3.10. Previous Models Revisited

Previous models have used the SMP in figure 3.7 as their PE behavioral description (some earlier "no think time" models even used a one reference state description of PE behavior). It might be said that the low level models have *assumed* that PE's do indeed behave (approximately) as ARP's. The waiting time approximations developed next supply some information regarding the applicability of this assumption. The SMP model reduction technique displays an *analytic* approach for finding the low level data previously acquired from instruction traces or simulations.

The reduction does not guarantee full temporal equivalence in that there may be timing differences between the ARP representation and the SMP representation, see figure 3.12 for an example. The equivalence is in the time limiting averages.

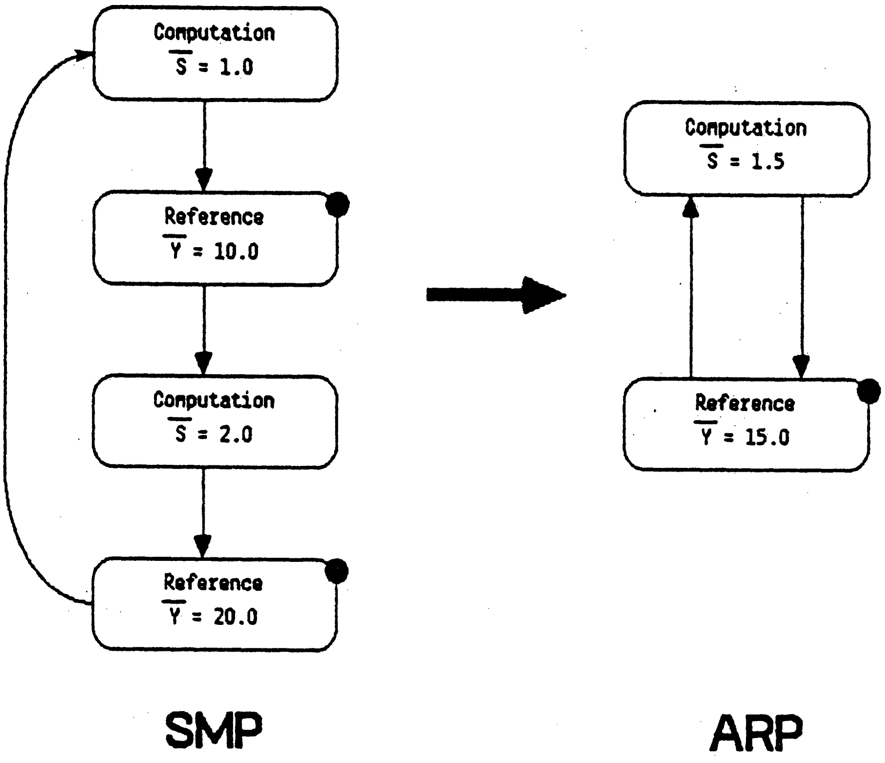


Figure 3.12 The SMP to ARP reduction results.

Consider computing the equivalent computation state sojourn time described in figure 3.7, again it is given by:

$$\overline{S^k_{p1}}' = \overline{T^k_p}(\Omega_{R_p} \rightarrow \Omega_{R_p}) \qquad (3.34\bullet)$$

Equivalent reference patterns are computed similarly, define η'_{pm} to be the probability that an arbitrary request emitted by PE p, as the system reaches steady-state, is for GRM m. Then

$$\eta'_{pm} = \sum_{s \in \Omega_{R_p}} \eta_{psm} \frac{\pi_{ps}}{\pi_{p\Omega_{R_p}}} \qquad (3.35\bullet)$$

Define $\overline{Y^k_{pm}}'$ to be the k^{th} moment of the equivalent connection time (the connection time for state 2 in figure 3.7), then:

$$\overline{Y^k_{pm}}' = \sum_{s \in \Omega_{R_p}} \overline{Y^k_{psm}} \frac{\pi_{ps}\,\eta_{psm}}{\sum_{j \in \Omega_{R_p}} \pi_{pj}\,\eta_{pjm}} \qquad (3.36\bullet)$$

where the product $\pi_{ps}\,\eta_{psm}$ represents the probability that the PE p request is from state s and GRM m is chosen when state s is entered. A sort of embedded GRM reference MC is apparent within the normal embedded MC formed by the transition points of the SMP. The state of the reference that caused the connection is unconditioned out of the statistics with the embedded reference chain's state s, GRM m reference occurrence frequency.

Notice that the quantities $\overline{S^k_{p1}}'$, η'_{pm}, and $\overline{Y^k_{pm}}'$ are low level quantities that may be measured at the hardware level. Since the above technique displays an *analytic* approach for finding low level data from program specifications, the SMP approach may be used to obtain low level data. The rates defined above may be rewritten using the defined ARP representation:

$$\lambda_{pm} = \frac{\eta'_{pm}}{\overline{C}'_p} \qquad (3.37\bullet)$$

λ_{psm} loses its meaning in the context of the ARP representation.

Expressing the performance measures as above with this ARP representation gives:

$$\phi_p = \frac{\overline{S}'_{p1}}{\overline{C}'_p}, \qquad \hat{\phi}_p = \frac{\overline{S}'_{p1}}{\overline{S}'_{p1} + \sum_m \overline{Y}'_{pm}\,\eta'_{pm}}$$

$$\rho_m = \sum_p \lambda_{pm} \overline{Y}'_{pm} \tag{3.38}$$

$$\overline{X}_m = \frac{1}{\lambda^{!}_m} \sum_p \lambda_{pm} \overline{Y}'_{pm}$$

High level computation state cycle times have no explicit representation in this PE activity representation.

Note that previous models have not dealt greatly with waiting time calculations explicitly. They have done a good job of predicting ρ_m (given the low level data) *using first moment analyses only,* but inverting ρ_m into $\overline{W}_{pm}$ (in an asymmetric case) is unsolved. Chapter 5 discusses the problems with the inversion of ρ_m into $\overline{W}_{pm}$. Due to this problem, new techniques have been devised for computing $\overline{W^{12}_{psm}}$ directly. They are covered next.

CHAPTER 4

WAITING TIME CALCULATIONS

This chapter details the remaining calculations required to solve the model equations. The solution technique that has been used to solve the model equations is a simple fixed point iteration scheme that uses sojourn time moments as variables of the solution scheme. It is depicted in figure 4.1. The relationship that maps $\overline{S_{ps}}$ into p_{ps} is given by (3.10). This chapter shows the relationships that map p_{ps}'s into $\overline{W^k_{psm}}$'s which in turn gives $\overline{S^k_{ps}}$'s from (3.6).

Before proceeding with the analysis an attractive alternative to computing waiting times analytically must be mentioned: a simulator could be developed to predict waiting times more accurately. The simulator would be written to operate at the level of the equivalent ARP PE behavior description. Initially, analytical approximations will be developed in this chapter. Using these results, requirements for a simulation solution will be described.

4.1. A State Space Reduction Technique

A technique has been developed that maps a given SMP (a PE state diagram) into an approximately equivalent two state representative process (the ARP). This is important as a computational efficiency aid, and is a result of the SMP model which serves to unify early models that used this two state representative process. Details of the reduction and properties are pursued next.

The reduction provides a technique for obtaining ARP parameters from a given SMP. The reduction of a given PE state diagram to a nearly equivalent ARP description allows economical computations to be done for performance measures including: PE

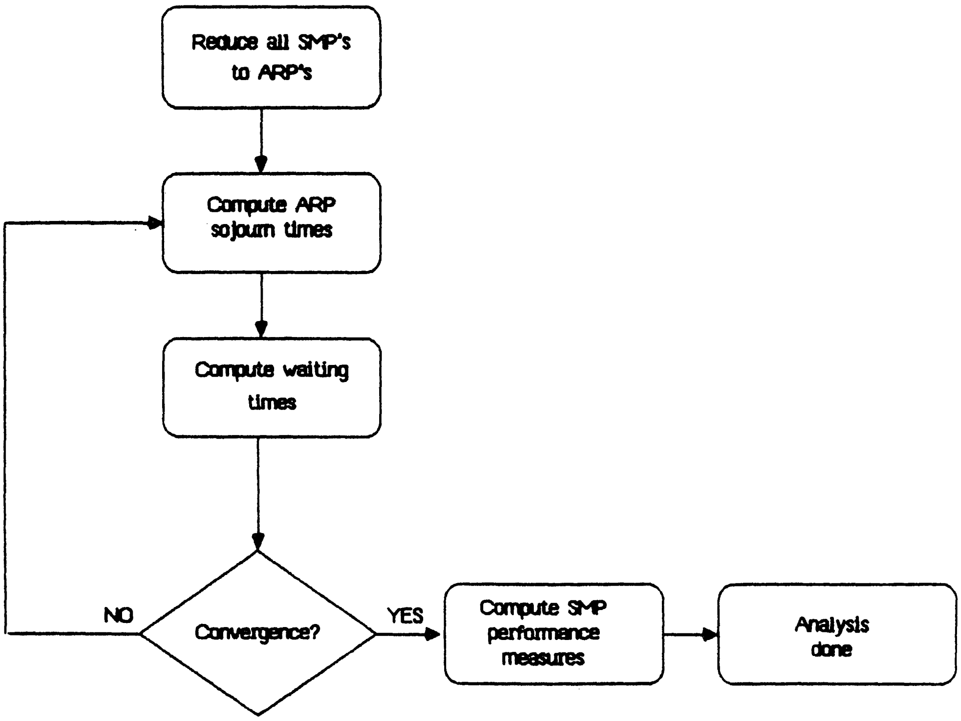

Reduce all SMP's to ARP's
Compute ARP sojourn times
Compute waiting times
Convergence?
NO
YES
Compute SMP performance measures
Analysis done

Figure 4.1 Simple fixed point solution scheme.

and GRM utilizations and coefficients of variation; connection time moments; request flow rates; GRM queue lengths; and to some extent, GRM waiting time moments.

There are two interpretations of the SMP to ARP reduction: they are computationally equivalent (i.e., using either the SMP or its ARP during the model solution will yield identical results); or waiting times are approximated as if they were state averages, that is, for each GRM reference a PE request perceives the waiting time averaged over all reference states. (This is analogous to the approximation that in queueing networks the source of a customer does not affect its waiting time statistics -- if a customer enters queueing station i from either station j or k it will see the same waiting time statistics regardless of its source of entry.) Consider the equivalence interpretation:

The equivalence in the above measures depends on the approximation that

$$\overline{W^k_{pm}} \simeq \overline{W^k_{psm}} \quad \textit{for all } s \in A_p' \qquad (\bullet)$$

holds to within the desired modeling accuracy. That is, that the first k moments of request queueing times are ***emission state independent.*** If this approximation is appropriate, then the reduction of a given PE state diagram to an equivalent (in PE and GRM class quantity prediction) ARP may be done. The reduction decreases the cost of subsequent computations. After PE and GRM class quantities have been computed, the high level timing quantities may be computed using SMP results from (3.6) and (3.12).

ARP computation time, connection time, and reference pattern statistics are computed from chapter 3 (3.34 - 3.36). These quantities are not subject to the applicability of the SMP to ARP reduction, they result from averaging SMP quantities to get the equivalent ARP values. The equivalence between the SMP to ARP reduction and $\overline{W^k_{psm}} = \overline{W^k_{pm}}$ is justified as follows:

(1) Assume $\overline{W^k_{psm}} = \overline{W^k_{pm}}$, then every reference state in the SMP may be partitioned into a series of two sub-states: one which accounts for queueing time -- this sub-state's sojourn time is $\overline{W^k_{psm}} = \overline{W^k_{pm}}$ and is emission state invariant hence it appears intact in the ARP reference state; and the connection sub-state -- which has a state average connection time CDF $Y_{pm}'(t)$. The ARP reference state sojourn time is a state average over all SMP reference states.

(2) Conversely, assume that the equivalent reduction from an SMP to ARP is possible. Then an arbitrary request "in the ARP" will experience a queueing delay given by $\tilde{W}_{pm}$. Hence an ***arbitrary*** request in the full SMP will experience $\tilde{W}_{pm}$ delay. ***All*** requests experience $\tilde{W}_{pm}$ delay independent of the state from which they were emitted in the full SMP. Therefore the delay $\tilde{W}_{pm}$ will be present in all of the SMP reference states so $\tilde{W}_{pm} = \tilde{W}_{psm}$.

It is certainly true that $\overline{W^k_{pm}}$ is not always $\overline{W^k_{psm}}$, it is reasonable to expect the approximation to hold in circumstances where GRM contention (interference) is not too great and PE's behave nearly independently. When PE's are independent, randomness in state occupation at time t is expected between processes due to queueing effects. That is, queueing effects serve to statistically separate GRM references from each other.

This tends to make $\tilde{W}_{psm}$ into $\tilde{W}_{pm}$ in the sense that correlation between GRM reference characteristics is not great (for a particular PE, there is certainly correlation in the request arrival streams at GRM's) unless, for example, two references to the same GRM by the same PE are arranged closely in time, this makes waiting time dependent on the state of emission. The effects of this approximation have not been found to be appreciable in that loop and execution times are typically composed of many queueing times, small differences have been found to be negligible. Simulations have shown (see chapter 6) that the approximation $\overline{W_{pm}^{k}} = \overline{W_{psm}^{k}}$ is reasonable in many circumstances. In fact, there has been no motivation to include more resolution by computing $\overline{W}_{psm}$ rather than $\overline{W}_{pm}$.

The effects of the approximation are all that is required. In that performance measures are the primary results of the model analysis, only as much resolution in computing waiting times as will yield acceptable performance measure predictions is required. Our primary interest is in the effects attributable to state independent waiting time moments.

Approaches for computing $\overline{W}_{pm}$ have been developed for the ARP description of PE behavior [BaS76, Smi74], unfortunately they have assumed that $\overline{W}_{pm}$ is independent of p, i.e. that all PE requests for GRM m experience the same queueing time statistics. It is too gross an approximation to be used realistically. While $\overline{W_{psm}^{k}} \simeq \overline{W_{pm}^{k}}$ may be considered to be approximate, $\overline{W_{psm}^{k}} \simeq \overline{W_{m}^{k}}$ is quite simply incorrect except in homogeneous cases. Previous approaches have done an accurate job of predicting ρ_m but most have not computed $\overline{W}_m$ from ρ_m (it may be seen that $\overline{W}_m$ -- it exists in the homogeneous process case -- may be approximated from ρ_m and ARP properties). Even if $\overline{W}_m$ is available no technique has been found that will invert $\overline{W}_m$ to get $\overline{W}_{pm}$. Further work here may be warranted. The following discussion illustrates appropriate computations of $\overline{W_{pm}^{1,2}}$.

Many analytic formulations are possible; the following have been used with a reasonable amount of success. Two techniques will be presented here to show the range of possibilities in this phase of the analysis. Both rely on the first two moments of computation state sojourn times and the first three moments of connection times.

It should be noted that the SMP model solution emphasizes the physical analysis of the ICN and GRM system. That is, it is most appropriate to perform a physical analysis of the resource system to obtain components of equation (3.6). Compare this to exponential or Markov chain based models of, for example, the same type of system where system "model state diagrams" may be used. For example, if the equivalent computation state and connection times are exponential, then a simple system Markov process may be defined [MaG81]. Using these techniques state space size is a significant problem. This *logical* analysis suffices in relatively simplified system configurations, but fails to be tractable in reasonably general, realistic systems.

Physical analysis of resource systems combined with the PE state diagram description allows the SMP model to be used in a tractable manner; assuming that the state space itself is small enough to be reduced to an "equivalent" ARP approximate representation used to perform the physical analysis. (Note that exponential PE descriptions are a special case of the SMP model.) The drawback of physical analysis is that the resource system may be reasonably complex and extensive knowledge about system behavior and appropriate modeling is required. Physical analysis is not quite as simple as solving a Markov chain problem. There is a tradeoff between analysis simplicity and tractability. Once physical analysis has been done, the use of its results is very economical when compared to logical analysis or complete simulation. Occasionally physical analysis may be formulated into a logical analysis problem (such as in analyzing an M/G/1 queue in isolation to obtain physical relationships), sometimes unacceptable simplifying assumptions must be made in the analysis formulation.

Because of its economical results, the physical analysis approach has been used here with much success. Physical analysis also gives explicit results or approximations which may not in general be possible when using logical analysis. Physical analysis also lends itself well to the substitution of simulated values required during the analysis. The physical analysis will be guided by previous results, intuition, and simulation data. The philosophy of using physical analysis with heuristic approximations used for previously unquantified effects is described in part in [Whi83]. Since we are dealing with analysis that has not yet been rigorously dealt with, heuristic approximation development supported by simulation validation is believed to be an appropriate approach.

4.2. PE Synchronization

During simulation studies it was noted that in some realistic circumstances PE's may self synchronize their request emission streams to obtain negligible or actually zero queueing times.

PE synchronization occurs in configurations where there is enough time between successive references by PE p such that -- with high probability -- there is enough time for several other PE's ($j \neq p$) to use GRM m. (Note that this would most probably be seen in configurations with highly discrete computation and connection times.) See figure 4.2 for a diagrammatic representation.

Note that exponential models of system behavior do not model synchronization (technically, for models which *assume* sojourn or connection time CDF's that have a density on an interval, stochastic system paths that exhibit synchronism occur with zero probability, although these stochastic paths are certainly in the sample space), they ignore the stochastic paths which display synchronization in an effort at achieve simplicity. These paths may easily occur though. An example is shown in chapter 6.

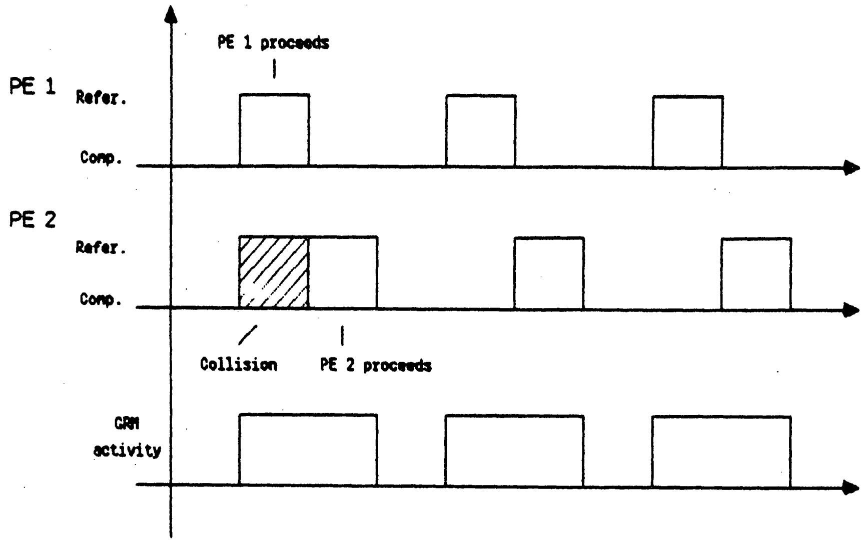


Figure 4.2 PE synchronization timing.

4.3. An Independent SMP Formulation for Waiting Times

First consider an approach based on the SMP independence approximation with emission time characteristics included. To calculate $W_{pm}(t)$, condition on the sequence of requests found in the GRM m queue, including the service position, at the time of a PE p request arrival. Define $\tilde{\Delta}_{pm}$ to be the sequence of (processor, state) pairs seen in GRM m by an arriving request from PE p referencing GRM m. An instance of $\tilde{\Delta}_{pm}$ might look like $\tilde{\Delta}_{pm} = \delta = ((p_1,s_1), (p_2,s_2), \cdots ,(p_k,s_k))$ where (p_i,s_i) is a tuple representing the PE number queued in position i and the state $s_i \in A_{p_i}^r$ in which PE p_i is waiting. p_1 is the PE using GRM m at the arrival time and p_k is the PE whose request arrived just prior to PE p's. Consider the FCFS queueing discipline from here on. Conditioning and unconditioning on $\tilde{\Delta}_{pm}$ gives:

$$W_{pm}(t) = \sum_{\delta} W_{pm}(t|\tilde{\Delta}_{pm} = \delta) Pr(\tilde{\Delta}_{pm} = \delta) \tag{4.1}$$

Again, because of the existence assumption (approximation) concerning $W_{pm}(t)$ we will assume that $Pr(\tilde{\Delta}_{pm} = \delta)$ exists for an arbitrary request arrival from PE p in the steady-state. (The time varying nature of the request emissions in actual system operation makes this an approximation.) Moments of $\tilde{W}_{pm}$ may be found from

$$\overline{W_{pm}^k} = \int_0^{\infty} t^k \, dW_{pm}(t) = \sum_{\delta} \overline{W_{pm|\tilde{\Delta}_{pm}=\delta}^k} \, Pr(\tilde{\Delta}_{pm} = \delta) \tag{4.2}$$

where $\overline{W_{pm|\tilde{\Delta}_{pm}=\delta}^k}$ is the k^{th} moment of the waiting time given the sequence δ was seen at the arrival point. This quantity is computed as

$$\overline{W_{pm|\tilde{\Delta}_{pm}=\delta}^k} = E\Big[\Big(\sum_{j=2}^{|\delta|} \tilde{Y}_{p_j s_j m} + \tilde{E}_{p_1 s_1 m}\Big)^k\Big] \tag{4.3}$$

where $\tilde{E}_{p_1 s_1 m}$ is a random variable representing the excess or residual connection time for the connection in progress at the arrival instant. Note that all random variables $\tilde{Y}_{p_j s_j m}$ and $\tilde{E}_{p_1 s_1 m}$ are independent by the assumed independence of PE's. $E_{p_1 s_1 m}(t)$ is in general difficult to obtain rigorously because: the GRM m service process is *not* renewal with CDF $Y_{p_1 s_1 m}(t)$; and the PE p arrival time is not Poisson (arriving requests do not see random arrival time statistics). There are, though, some approximate guidelines that may be used in formulating $\bar{E}_{p_1 s_1 m}$.

4.3.1. The Excess Connection Time

If service processes form renewal processes with service time $\tilde{Y}_{p_1 s_1 m}$ (i.e., that PE p_1 uses GRM m repeatedly) and PE p requests arrive at random (Poisson) times then $\overline{E}_{p_1 s_1 m} \simeq \dfrac{\overline{Y^2_{p_1 s_1 m}}}{2\overline{Y}_{p_1 s_1 m}}$ from residual life theory [HeS82, Ros70]. Conversely, it has been noted that in circumstances where PE's self synchronize $\overline{E}_{p_1 s_1 m} = 0$ is possible even when $\overline{Y^2_{p_1 s_1 m}} > 0$, $\overline{Y}_{p_1 s_1 m} > 0$. Delay until section 4.3.3 further discussion of $\overline{E}_{p_1 s_1 m}$. Consider next the computation of $Pr(\tilde{\Delta}_{pm} = \delta)$.

4.3.2. Computation of $Pr(\tilde{\Delta}_{pm} = \delta)$

In general $Pr(\tilde{\Delta}_{pm} = \delta)$ depends on all stochastic processes in the system in that it is influenced by the state of PE's $j \neq p$ at the emission time. The emission time characteristics are important but are inherently too complicated to handle rigorously, therefore approximations will be made. PE's $j \neq p$ are, in turn, are influenced by their previous GRM references which include effects from PE p itself. Hence PE p affects its own arrival point statistics. Due to the complexity of including feedback effects and the computational cost of evaluating (4.1) and (4.3) over all sequences δ, the approach indicated by (4.2), although valid, will not be pursued here. See [MaM82] for an example of the summations involved. An approximate approach will be used that takes PE's to be independent and the emission time to be a random time for the purposes of computing arrival point queue lengths, a heuristic correction factor will then be applied to reflect PE p's influence on its own arrival point queue lengths. Consider the first moment of waiting times, the central approximation is:

$$\overline{W}_{pm} \simeq \overline{N}_{pm} \overline{X}_m(p)(1 + \rho_{pm}) + (1-\alpha_{pm})\overline{E}_{pm} \qquad (4.4\bullet)$$

Where

$\overline{E}_{pm}$ is the mean excess connection time similar to above averaged over all PE's $j \neq p$ and their reference states (we are using the approximation that arrival point statistics are independent of the emission state), that is, it represents the excess connection time for an *arbitrary* request arriving at GRM m from PE p,

$\overline{X}_m(p)$ is the mean connection time averaged over PE's $j \neq p$,

α_{pm} is the probability that GRM m is free at the PE p request arrival time, and

$\overline{N}_{pm}$ is the mean GRM m queue length seen by an arriving request from PE p, as computed without including history effects due to PE p.

The quantity $1 + \rho_{pm}$ is used to approximate the effects of PE p on itself attributable to its previous use of GRM m. Hence $\overline{N}_{pm}(1 + \rho_{pm})$ represents an approximate first moment of the queue length seen by an arriving request from PE p. Since $1 + \rho_{pm}$ is a heuristic correction factor it is an approximation that requires testing. ρ_{pm} is computed as

$$\rho_{pm} = \lambda_{pm} \overline{Y}'_{pm} = \sum_{s \, \epsilon \, A'_p} \lambda_{psm} \overline{Y}_{psm} \tag{4.5•}$$

from the ARP or SMP values. $\overline{X}_m(p)$ is computed as

$$\overline{X}_m(p) = \frac{1}{\lambda_m(p)} \sum_{j \neq p} \lambda_{jm} \overline{Y}'_{jm} = \frac{1}{\lambda_m(p)} \sum_{j \neq p} \rho_{jm} = \frac{\rho_m(p)}{\lambda_m(p)} \tag{4.6}$$

And

$$\overline{X^k_m}(p) = \frac{1}{\lambda_m(p)} \sum_{j \neq p} \lambda_{jm} \overline{Y^k_{jm}}'$$

where $\rho_m(p)$ is ρ_m without PE p's contribution: $\rho_m(p) = \rho_m - \rho_{pm}$.

α_{pm} is approximated with general time quantities:

$$\begin{aligned} \alpha_{pm} &= Pr(\text{no PE } j \neq p \text{ is using GRM } m \text{ at the emission time } t) \\ &\simeq \prod_{j \neq p} Pr(\text{PE } j \text{ is not using GRM } m \text{ at } t) \end{aligned}$$

$$\begin{aligned} &Pr(\text{PE } j \text{ is not using GRM } m \text{ at } t) \\ &\qquad = 1 - Pr(\text{PE } j \text{ is using GRM } m \text{ at } t) \\ &\qquad \simeq 1 - \frac{\eta'_{jm}(\overline{W}_{jm} + \overline{Y}'_{jm})}{\overline{C}'_j} \end{aligned}$$

So

$$\alpha_{pm} \simeq \prod_{j \neq p} \left[1 - \frac{\eta'_{jm}(\overline{W}_{jm} + \overline{Y}'_{jm})}{\overline{C}'_j} \right] \tag{4.7}$$

where $\overline{C}'_j$ is the mean cycle time for PE j's equivalent ARP. Formally, we are using the approximation that either t is a random time or t is such that PE p emits a request at a lattice point of all other PE ARP cycle times. $\overline{C}_j'$ is given by:

$$\overline{C}_j' = \overline{S}_{j1}' + \sum_m (\overline{W}_{jm} + \overline{Y}_{jm}')\eta_{jm}' \qquad (4.8\bullet)$$

Finally consider $\overline{N}_{pm}$; again the approach is to take the emission time to be a time-limiting general time. To reduce the complexity of computing $Pr(\tilde{\Delta}_{pm} = \delta)$ (even without PE p's own influence) an averaging reduction technique will be employed. Define $\overline{\overline{W}}_m(p)$ to be the mean of the first moment of waiting times, averaged over all PE's but p,

$$\overline{\overline{W}}_m(p) = \sum_{j \neq p} \left(\frac{\lambda_{jm}}{\lambda_m(p)} \right) \overline{W}_{jm} = \frac{1}{\lambda_m(p)} \sum_{j \neq p} \lambda_{jm} \overline{W}_{jm} \qquad (4.9)$$

We are building a unified mean waiting time to be used in writing simple expressions. The quantity $\overline{\overline{W}}_m(p)$ is not a performance measure in itself but is an internal variable used in the evaluation of $\overline{W}_{pm}$. Define q_{pm} to be the probability that a request from another PE is in GRM m at an emission time for PE p referencing GRM m. That is, it is an average over all PE's $j \neq p$:

$$q_{pm} = \sum_{j \neq p} \left(\frac{\lambda_{jm}}{\lambda_m(p)} \right) \left(\frac{\overline{W}_{jm} + \overline{Y}_{jm}'}{\overline{C}_j'} \right) \eta_{jm}' \qquad (4.10)$$

If these averaged quantities $\overline{\overline{W}}_m(p)$, and q_{pm} are valid (they are most accurate in the homogeneous PE case), then the arriving request sees a binomial distribution of requests in GRM m. This approach gives an approximate closed form evaluation for $Pr(|\tilde{\Delta}_{pm}| = \delta)$:

$$Pr(|\tilde{\Delta}_{pm}| = \delta) \simeq \binom{|Q_{pm}|}{\delta} \delta! \; q_{pm}^{\delta} (1 - q_{pm})^{|Q_{pm}| - \delta} \times \left(\frac{\overline{X}_m(p)}{\overline{\overline{W}}_m(p) + \overline{X}_m(p)} \right) \left(\frac{\overline{\overline{W}}_m(p)}{\overline{\overline{W}}_m(p) + \overline{X}_m(p)} \right)^{\delta - 1} \qquad (4.11)$$

A set of counted requestors is introduced as Q_{pm} :

$$Q_{pm} = \left\{ j \neq p \colon \rho_{jm} \geq \frac{\rho_m(p)}{|\#_{pm}|} \right\} \qquad (4.12\bullet)$$

and $\#_{pm}$ is a set of feasible requestors:

$$\#_{pm} = \{j \neq p: \lambda_{jm} > 0\} \qquad \textbf{(4.13•)}$$

$\#_{pm}$ consists of requestors that are feasibly seen by an arriving request from PE p referencing GRM m. The set Q_{pm} is a set of requestors that are to be counted as in the separated queueing system composed of PE's $j \neq p$ and GRM m. This "countability" indicates the fact that some PE's may influence GRM m more than other PE's, hence counting all requestors equally in Q_{pm} would lead to inaccurate results. The above thresholding scheme is based on an "average threshold" value. Note that in a symmetric case where all PE's behave identically $|\#_{pm}| = P - 1$ and $\rho_m(p) = \rho_{pm}(P-1)$ so $|Q_{pm}| = P - 1$ as would be expected. This thresholding scheme has been found to work reasonably well, see the asymmetric example in chapter 6. Other thresholding schemes having lower thresholds (and hence higher $|Q_{pm}|$) have been found to overestimate $\overline{W}_{pm}$ in asymmetric cases, hence thresholding is important within the approximations developed here.

The quantity $\begin{pmatrix} |Q_{pm}| \\ \delta \end{pmatrix} \delta!$ is the number of ways that δ requests from Q_{pm} may be placed in the GRM m queue and service positions.

$(1 - q_{pm})^{|Q_{pm}| - \delta}$ is the probability that $|Q_{pm}| - \delta$ requests are not in GRM m at the arrival time.

$$q_{pm}^{\delta} \left(\frac{\overline{X}_m(p)}{\overline{\overline{W}}_m(p) + \overline{X}_m(p)} \right) \left(\frac{\overline{\overline{W}}_m(p)}{\overline{\overline{W}}_m(p) + \overline{X}_m(p)} \right)^{\delta - 1} \qquad (4.14)$$

is the probability that δ request are in GRM m at the arrival time. $\delta - 1$ requests are in queue with probability

$$\left(q_{pm} \left(\frac{\overline{\overline{W}}_m(p)}{\overline{\overline{W}}_m(p) + \overline{X}_m(p)} \right) \right)^{\delta - 1} \qquad (4.15)$$

(each reference for GRM m is composed of two phases -- waiting and service, the phases are occupied with fractions $\overline{\overline{W}}_m(p)/(\overline{\overline{W}}_m(p)+\overline{X}_m(p))$ and $\overline{X}_m(p)/(\overline{\overline{W}}_m(p)+\overline{X}_m(p))$ respectively given that the PE's have requests in GRM m); and one is in service with probability $q_{pm}\overline{X}_m(p)/(\overline{\overline{W}}_m(p) + \overline{X}_m(p))$. Since only one request may be in service, the other $\delta - 1$ are excluded from this phase of their reference state sojourn. Then

$$\overline{N}_{pm} \simeq \sum_{\delta=1}^{|Q_{pm}|} (\delta - 1)Pr(|\tilde{\Delta}_{pm}| = \delta) \qquad \textbf{(4.16•)}$$

4.3.3. Characteristics of the Mean Excess Connection Time

A simple heuristic function has been used to approximate $\overline{E}_{pm}$ by assuming the results of a renewal service process being examined at a random time, corrected with a multiplying factor for the application. The form used is:

$$\overline{E}_{pm} = \frac{\overline{X_m^2}(p)}{2\overline{X}_m(p)} f_{pm} \qquad (\bullet)$$

$\overline{X_m^2}(p)/(2\overline{X}_m(p))$ is the approximate excess connection time when $\rho_m(p) = 1$. f_{pm} is used to correct this value when $\rho_m(p) < 1$. (Note that f_{pm} might be more properly depicted as f_{psm} but we ignore emission state dependence to reduce complexity.)

Although introducing a multiplication factor is approximate, it is an attempt to include effects that are computationally expensive to handle in a more rigorous manner. Since it is believed that models should provide economical, approximate predictions, simple approximations have been developed. The evaluation is guided by quantitative and qualitative aspects of queueing theory.

The general shape of f_{pm} used is shown in figure 4.3. The ratio of λ_{pm} to $\lambda_m^{\,!}$ is a measure of the frequency with which PE p uses GRM m relative to all requestors. Another possibility is the use of $\lambda_{pm}/\lambda_m(p)$ as an indicator of PE p use relative to other requestor usage. When this indicator is near zero, PE p rarely uses GRM m -- the mean time between uses is large -- and hence PE p's interarrival time CDF "approaches" an exponential interarrival time statistic so we set $f_{pm} = 1$.

As the usage frequency (call it u_{pm}) increase from zero (it is bounded by one when it is taken to be $\lambda_{pm}/\lambda_m^{\,!}$), f_{pm} may decrease because of synchronization effects. The amount of decrease is related to the coefficient of variation of request interarrival times at GRM m. As u_{pm} increases coupling between request streams increases. Hence determinism and correlation between inter-reference times in the input stream increases with u_{pm}. At the boundary (i.e., when the stream becomes fully deterministic). When $V_{M_m}^2 = 0$ a PE p request in the arrival stream sees a constant residual connection time, otherwise $V_{M_m}^2 > 0$. In the 2×2 case this occurs when requests alternate due to previous

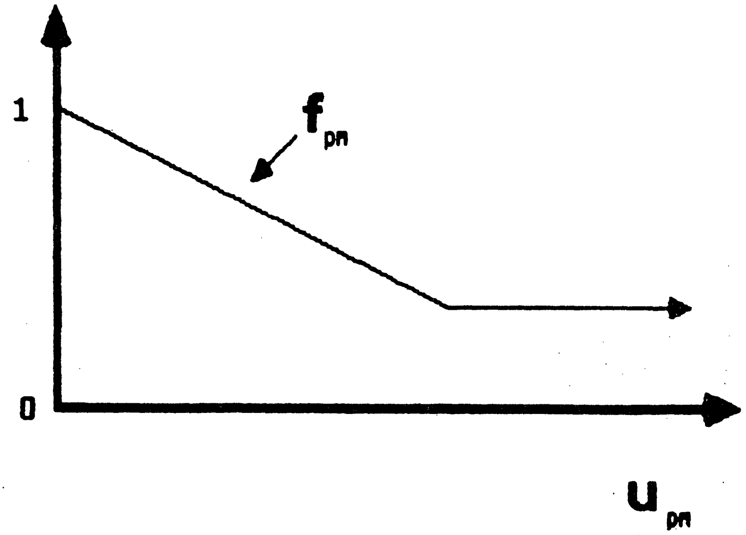


Figure 4.3 Linear interpolation for f_{pm}.

collisions: the residual connection time is zero. Hence when $u_{pm} \simeq 1$ and $V_{M_m}^2 \simeq 0$, no residual will be seen so the form for f_{pm} (using a linear interpolation between endpoints) used is:

$$f_{pm} = \begin{cases} (V_{M_m}^{1,2}(p) - 1)u_{pm} + 1 & u_{pm} \leq 1,\ V_{M_m}^{1,2}(p) \leq 1 \\ V_{M_m}^{1,2}(p) & u_{pm} > 1,\ V_{M_m}^{1,2}(p) \leq 1 \\ 1 & V_{M_m}^{1,2}(p) > 1 \end{cases} \qquad (4.17\bullet)$$

or

$$f_{pm} = \begin{cases} (V_{M_m}^{1,2} - 1)u_{pm} + 1 & u_{pm} \leq 1,\ V_{M_m}^{1,2} \leq 1 \\ V_{M_m}^{1,2} & u_{pm} > 1,\ V_{M_m}^{1,2} \leq 1 \\ 1 & V_{M_m}^{1,2} > 1 \end{cases} \qquad (4.18\bullet)$$

Depending on whether the PE p request stream is included in the measurement of the amount of determinism in the GRM m request stream. Coefficients of variation to the first or second power might be used. Experimental data has shown which choices are most appropriate. In practice, only two of the possible eight cases have been found to be useful: the one that generates the smallest f_{pm} ; and the one that generates the largest. Call these cases two cases 1 and 2 respectively.

And

$$u_{pm} = \frac{\lambda_{pm}}{\lambda_m(p)} \qquad (4.19\bullet)$$

or

$$u_{pm} = \frac{\lambda_{pm}}{\lambda_m^{'}} = \frac{\lambda_{pm}}{\lambda_m(p) + \lambda_{pm}} \qquad (4.20\bullet)$$

Again depending on which is chosen to represent the usage frequency.

4.4. A G/G/1 Approach to Waiting Time Calculations

As an alternative to "probabilistic" analyses for evaluating $\overline{W}_{pm}$, consider the use of a finite customer queueing network model of the system. Finite customer network analysis has typically been limited to exponential service times or approximations based on Jackson type networks. An analysis of queueing networks based on the first *two* moments of service times was described in [Kue79], unfortunately [Kue79] yields inaccurate results for small customer populations as seen in the networks considered here (i.e., a small number of PE's, say less than 32). The basic approach from [Kue79] has been enhanced with heuristic approximations for the system considered here. The approach might be generally useful in small population queueing networks but has not been tested generally.

The G/G/1 approximation for computing $\overline{W}_{pm}$ is based on viewing the GRM m queue from the viewpoint of PE p. It is based in part on the idea that in a Jackson type of finite customer queueing network arrival point queue statistics are those obtained as the steady state solution for a system without the arriving customer (i.e., a population decreased by one, see [SeM81]). Here a similar but not equivalent approach is taken which separates PE p requests from a subsystem composed of PE's $j \neq p$ and GRM m. That is, a PE p request sees a subsystem composed of all other PE's and GRM m. GRM m then behaves approximately as a G/G/1 queue with a renewal input process formed by PE's $j \neq p$ and service times $\tilde{X}_m(p)$. This is asymptotically most accurate as the mean time between GRM m references by PE p tends to infinity (i.e., PE p contributes a negligible amount of load to GRM m). The GRM m input process interarrival time moment calculations are based on [Kue79] (with corrections for case 1 detailed in the appendix) and a G/G/1 mean waiting time approximation from [KrL76] as stated in [Kue79]. Again the mean waiting time must be modified to account for the finite number of requests that may be in GRM m at an arrival point *(at most* $\#_{pm}$*)*, and PE p's contribution to its own mean waiting time. Note that the independent SMP calculation above was inherently a finite customer approach whereas an uncorrected G/G/1 approach is inherently an infinite customer model. The G/G/1 calculations follow:

For a G/G/1 queue with mean arrival rate α, coefficient of variation of the interarrival time β, mean service time γ, and the coefficient of variation of service time δ, the mean waiting time is approximately [KrL76]:

$$\overline{W} = \gamma \frac{\alpha\gamma}{2(1-\alpha\gamma)} (\beta^2 + \delta^2)\, u(\alpha,\gamma,\beta,\delta) \tag{4.21}$$

Where

$$u(\alpha,\gamma,\beta,\delta) = \begin{cases} \exp\left\{ -\dfrac{2(1-\alpha\gamma)}{3\alpha\gamma} \dfrac{(1-\beta^2)^2}{\beta^2+\delta^2} \right\} & \beta < 1 \\[2ex] \exp\left\{ -(1-\alpha\gamma) \dfrac{\beta^2 - 1}{\beta^2 + 4\delta^2} \right\} & \beta \geq 1 \end{cases}$$

To account for the finite number of requests in the system, consider using the product $\overline{W}g_{pm}$ where g_{pm} represents an approximate function used to limit $\overline{W}$ and is a function of system size and utilizations. Asymptotic values for g_{pm} are:

$$\left| g_{pm} \right|_{|Q_{pm}| = 0} = 0,\ \lim_{|Q_{pm}| \to \infty} g_{pm} = 1,\ g_{pm} \downarrow \textit{as } \rho_m(p) \uparrow,\ g_{pm} \uparrow \textit{as } \rho_m(p) \downarrow,\ 0 \leq g_{pm} \leq 1. \tag{4.22•}$$

When there are no other requestors in the system, $g_{pm} = 0$. As the number of other requestors grows, g_{pm} increases to one. As $\rho_m(p)$ increases, GRM utilization by other requestors increases, hence the potential for synchronization increases from increased coupling, therefore g_{pm} decreases; a small g_{pm} creates a small waiting time. Due to the general nature of the asymptotic limits, there are an infinite number of choices for g_{pm}. One which has been found to work reasonably well is:

$$g_{pm} = (1 - e^{-|Q_{pm}|})^{\rho_m(p)} \tag{4.23•}$$

This is an approximation that has been found to work sufficiently well with the following modifications that account for dependence on ρ_{pm}:

$$\overline{W}_{pm} \simeq min \{ \overline{X}_m(p)^*|Q_{pm}|, \\ \overline{X}_m(p)^*E_1(p,m)^*E_2(p,m)^*g_{pm}{}^*h_{pm} \} \qquad (4.24\bullet)$$

Where

$$E_1(p,m) = \frac{V_{M_m}^2(p) + V_{S_m}^2(p)}{2(1 - \rho_m(p))}$$

$$E_2(p,m) = u(\lambda_m(p), \overline{X}_m(p), V_{M_m}(p), V_{S_m}(p))$$

$V_{M_m}'(p)$ is the coefficient of variation of the time between request arrivals at GRM m without PE p's contribution. $V_{S_m}(p)$ is the coefficient of variation of the service time (connection time) for GRM m without PE p's contribution. h_{pm} is a history dependence factor. If GRM m were an infinite customer subsystem or many PE's or requests in the system, then $h_{pm} = \rho_m(p)$ would be used (contributions due to PE p may be insignificant), to include the most straightforward dependence on ρ_{pm} the following expression for h_{pm} has been used:

$$h_{pm} = \rho_m(p) + \rho_{pm} = \rho_m \qquad (4.25\bullet)$$

The inclusion of ρ_{pm} represents an approximation for history effects. This is analogous to the $(1 + \rho_{pm})$ multiplier in (4.4). Notice that again the first two moments of computation and reference state sojourn times appear, although here they arise in the queueing equations inherently. In the independent SMP approach they arose when the excess connection time was considered.

4.5. Second Moment Calculations

The second moments of waiting times are required in computing V_{M_m} above. Second moment waiting time calculations are less critical than first moment (assuming the first moments of measures are of primary interest) and as such a simple M/G/1 approach will be used. It is motivated by superposition limiting theorems as approximations [Cin72]. Consider using the Takacs M/G/1 waiting time equation with the finite customer factor g_{pm}:

$$\overline{W_{pm}^2} = \left[2(\overline{W}_{pm})^2 + \frac{\lambda_m(p)\overline{X_m^3}(p)}{3(1 - \rho_m(p))} \right] g_{pm} \qquad (4.26\bullet)$$

See [MaM82] for a similar first moment approximation that was based on the fact that the superposition of sparse renewal processes leads to a Poisson process [Cin72]. It is most appropriate for large systems where the input process at GRM queues tends to be a Poisson process. In fact it has been noted that *even small systems have a strong Poisson* tendency under many circumstances if the coefficient of variation is used to measure the approximate "distance" to a Poisson process. An improvement in (4.26) may be achieved by taking into account the finiteness of the system and coupling effects as in the first moment computations.

4.6. Coefficients of Variation

The existence of the coefficient of variation requires that processes be renewal. This may not be true in an actual system but will be assumed to be true in order to derive approximate values, this approximation is often used in queueing network analysis [Kue79, MuM82b, MuM82c].

To obtain the coefficients of variation required throughout the calculations, the relationships given here have been used:

$$V_{S_m}^2(p) = \frac{\overline{X_m^2}(p) - (\overline{X}_m(p))^2}{(\overline{X}_m(p))^2} \qquad (4.27\bullet)$$

$$V_{M_m}^2(p) = \underset{j \neq p}{\circledast} V_{jm}^2 \qquad (4.28\bullet)$$

$\circledast$ denotes the superposition operator on V_{pm}^2 defined by formulations in [Kue79]. V_{pm}^2 is the coefficient of variation squared for the time between GRM m references by PE p. V_{pm}^2 is related to $V_{P_p}^2$ as follows [Kue79]:

$$V_{pm}^2 = 1 - \eta'_{pm}(1 - V_{P_p}^2) \qquad (4.29\bullet)$$

Where $V_{P_p}^2$ is computed using cycle time moments in the equivalent ARP:

$$V_{P_p}^2 = \frac{\overline{C_p^{2'}} - (\overline{C}_p')^2}{(\overline{C}_p')^2} \qquad (4.30\bullet)$$

$\overline{C}_p'$ is given by (4.8) and $\overline{C_p^{2'}}$ is given by:

$$\overline{C_p^{2'}} = \sum_m E\left[(\tilde{S}'_{pl} + \tilde{W}_{pm} + \tilde{Y}'_{pm})^2\right]\eta'_{pm} \qquad (4.31\bullet)$$

Since for a particular PE ARP cycle these random variables are independent:

$$\begin{aligned}\overline{C_p^2}' &= \overline{S_{p1}^2}' + \sum_m \overline{W_{pm}^2}\eta_{pm}' + \sum_m \overline{Y_{pm}^2}' \eta_{pm}' \\ &+ 2\overline{S}_{p1}' \sum_m \overline{W}_{pm}\eta_{pm}' + 2\overline{S}_{p1}' \sum_m \overline{Y}_{pm}'\eta_{pm}' + 2\sum_m \overline{W}_{pm}\overline{Y}_{pm}'\eta_{pm}'\end{aligned} \tag{4.32•}$$

$$V_{M_m}^2 = V_{M_m}^2(p) \circledast V_{pm}^2 . \tag{4.33}$$

Note that the second moment of queueing times is very approximate, this leads to overestimation of the second moment of cycle times. The Bernoulli loop approximation will also over estimate the second moment of transition times, these two facts combine to over estimate V_{P_p}. This is a rough interpretation, so that in fact it may not always hold depending on the solution point reached in solving the equations. That is, because of the complexity of the equations it should be expected that rough statements are occasionally violated due to asymmetry, side effects of one variable on another, etc.

4.7. A Solution Procedure

Since interest was directed toward model development and analysis and not toward an elegant solution technique, a simple fixed point iteration scheme has been used to solve the model equations. The fixed point scheme used is:

(1) Initialize waiting time moments to zero. This is actually not that important, the solution values obtained are independent of reasonable initial values.

(2) Solve $\pi_p = \pi_p \mathbf{P}_p$ for all p, $1 \le p \le P$.

(3) Reduce all given SMP's to equivalent ARP's by using (3.34, 3.35, and 3.36).

(4) For all PE's, compute the mean inter-reference times $\overline{C}_p'$ from (4.8).

(5) For all PE's, compute the coefficient of variation of the request streams, given by (4.32, 4.30, 4.29, and 4.28).

(6) For all PE's and GRM's, compute λ_{pm} from (3.37).

(7) For all PE's and GRM's, compute $\lambda_m(p) = \sum_{j \neq p} \lambda_{jm}$.

(8) For all PE's and GRM's, compute $\overline{X_m^{1,2,3}}(p)$ and $\rho_m(p)$ from (4.6). Compute the first two moments of waiting times from: (4.4, 4.5, 4.7, 4.17 or 4.18, 4.19 or 4.20, 4.9, 4.10, 4.13, 4.12, 4.11, 4.16, 4.26) for the SMP analysis technique; or (4.13, 4.12, 4.27, 4.23, 4.24, 4.25, 4.26) in the case of the G/G/1 analysis. This waiting time calculation must be done in a single phase, that is, a Jacobian update of the waiting time

matrices $\overline{\mathbf{W}}$ and $\overline{\mathbf{W}^2}$ is used. The Gauss-Seidel update technique may cause the solution to fail.

(9) Go back to step 4 until convergence is met.

(10) For all GRM's, compute arrival stream coefficients of variation $V_{M_m}^2$ from (4.33).

(11) Compute sojourn times from (3.6).

(12) Compute p_{ps} from (3.10).

(13) Compute λ_m^i from (3.22).

(14) Compute $\overline{X_m^{1,2}}$ from (3.25).

(15) Compute ρ_m from (3.26) (and potentials).

(16) Compute ϕ_p from (3.15) (and potentials).

(17) Compute $\overline{C}_{pss}$ from (3.12).

In general, the solution procedure may be run several times to obtain bounds described in chapter 5.

The iteration loop is that typically seen in fixed point vector iteration schemes, convergence is defined to occur when there is little difference in pertinent variables from the n^{th} iteration to the $n{+}1^{st}$. If convergence is not found within a given maximum count, then the solution has failed. It has been noted that in cases where convergence in either the SMP queueing analysis or the G/G/1 analysis occurs, it typically takes fewer than 15 iterations. The G/G/1 solution does not always converge, this will be demonstrated in chapter 6.

4.8. Simulation Analysis of Waiting Times

As an alternative to the analytic prediction of waiting time moments, consider a combination of model analysis and simulation solution. In particular, it is feasible to use the SMP to ARP reduction technique to form a set of simulation parameters. A simulation may be invoked at the ARP level of system operation to obtain predictions of waiting times. These predictions may then be used in the model equations to complete the original program state cycle time moment analysis. The simulation approach is particularly attractive when more accurate results are required than analytic approaches may provide. The reduction to an ARP forms an intermediate level where simulation is not prohibitively expensive and the results may be quite accurate.

During simulation, the CDF's $S_{p1}'(t)$, $Y_{pm}'(t)$ will be required for random variable sampling. Hence representative CDF's need to be derived. An appropriate approach is to note *from the above two waiting time analyses* the number of moments that seem to be required in representative CDF's if the simulation analysis for waiting times is to be at least as accurate as the analytic approximations. From the analysis it may be seen that the first two moments of the equivalent computation state sojourn time ($\overline{S_{p1}^{1,2}}'$) must be matched to the first two moments of the representative CDF. Similarly, the first three moments of equivalent ARP connection times ($\overline{Y_{pm}^{1,2,3}}'$) must match the first three moments of the representative connection time CDF chosen. ([Fre82] suggest moment matching to approximate GI/G/1 waiting times.) Example calculations follow that may clarify the point.

Since only the first two moments of the computation state sojourn time arise in the queueing analysis, two moment matching may be used with a CDF function chosen to represent the computation state sojourn time CDF. [Kue79] contains a representation for CDF's which match the first two given moments. The first two moments of each PE's equivalent computation state sojourn time would be obtained from the SMP to ARP reduction process previously described.

A three moment representation CDF for $Y_{pm}(t)$ will be described as an example of the possibilities. Recall that $\overline{Y_{pm}^{1,2,3}}'$ are derived from the SMP to ARP reduction process. The equivalent delay time CDF that will be described is shown in figure 4.4. Others might be chosen, generally these should be chosen to closely replicate the natural behavior of the system operation under consideration.

$$Y_{pm}(t) = \begin{cases} 0 & t \le \Delta \\ 1-(\,pe^{-\varsigma(t-\Delta)} + (1-p)e^{-k\varsigma(t-\Delta)}\,) & t > \Delta \end{cases}$$

k is chosen by the model user and should reflect the amount of determinism expected in connection times. Δ, ς and p are found from matching the first three moments as follows:

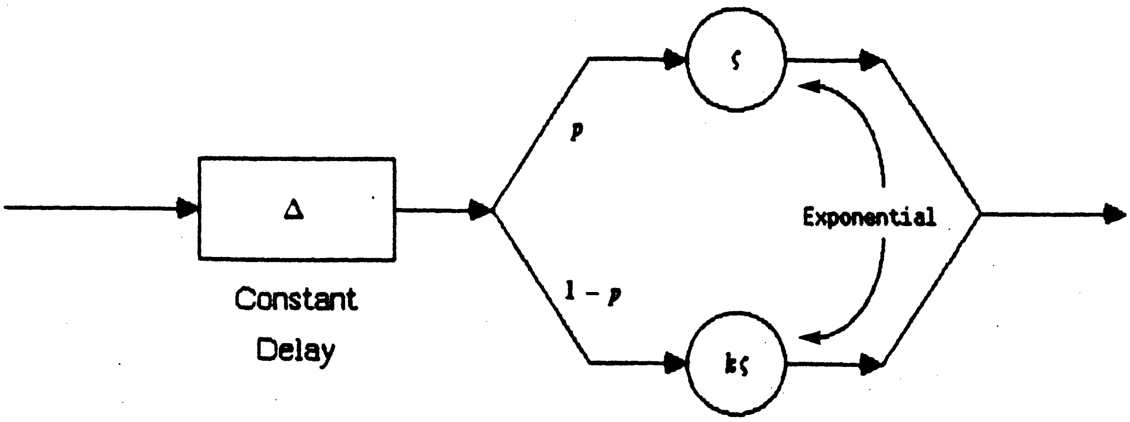


Figure 4.4 Three moment representative timing.

$$\overline{Y}_{pm} = \int_0^\infty t\,dY_{pm}(t)$$

$$= \Delta + \frac{p}{\varsigma} + \frac{1-p}{k\varsigma} \qquad \varsigma = \infty,\ p = 1 \ \ \mathit{if}\ \overline{Y_{pm}^2} = (\overline{Y}_{pm})^2,\ \ \overline{Y_{pm}^3} = (\overline{Y}_{pm})^3$$

$$\overline{Y_{pm}^2} = p\Delta^2 + \frac{2p\Delta}{\varsigma} + \frac{2p}{\varsigma^2} - (1-p)\frac{k\varsigma\Delta^3}{3}$$

$$\overline{Y_{pm}^3} = p\Delta^3 + \frac{3p\Delta^2}{\varsigma} + \frac{6p\Delta}{\varsigma^2} + \frac{6p}{\varsigma^3} - (1-p)\frac{k\varsigma\Delta^4}{4}$$

Which establishes Δ,ς, and p in terms of $\overline{Y_{pm}^{1,2,3'}}$ and k.

The simulation approach for determining waiting times is still subject to the higher moment approximation present in the SMP to ARP reduction process and the Bernoulli loop approximation. Although the simulation technique has not been developed further it is believed that it would yield results at least as accurate as the analytic approximations described.

4.9. Conclusions

This chapter contains an appropriate physical analysis for a crossbar based system which operates according to the specifications given in chapter 1. The results presented also have applications in small population, reasonably general queueing network analysis. The idea of physical analysis allows different ICN behaviors to be modeled by doing the appropriate physical analysis for ICN queueing times without changing the basic model framework.

CHAPTER 5

GENERAL RELATIONSHIPS AND BOUNDS

This chapter describes relationships and bounds that provide further information about system operation and insight into system characteristics.

Bounds on performance measures that depend on first moments of connection and computation times will be described. Bounds dependent on first moments are important when highly accurate behavior prediction is not required, or when an in depth analysis of connection and computation times is not feasible. The bounds have been found to work reasonably well in practice.

The mean outside observer queue length, $\bar{N}_m$, is also discussed. Interesting relationships concerning waiting times are derived, they indicate that for some system configurations, resource queueing *must* exist (i.e. there may not exist a perfect synchronization scheme). Simple sensitivity analysis is described which demonstrates insensitivity of GRM utilizations to changes in waiting times. A formulation of derivative equations relating system quantities is also discussed.

5.1. Bounds

Consider using first moment information in bounding first moment measures and utilizations. State sojourn times will be bounded by bounding waiting times. The resultant bounds on sojourn times may then be used to bound performance measures. Three bounding schemes are readily apparent:

(1) First consider lower bounding sojourn times. An obvious lower bound on sojourn times (with the first moments for sojourn and connection time parameters assumed available) is obtained by *setting all waiting times to zero,* assuming all computation

and connection times to be deterministic and then computing the measure values desired. This procedure yields hard bounds in that the bounds obtained (upper bounds on ρ_m, lower bounds on mean cycle times, upper bounds on ϕ_p, and lower bounds on V_{P_p} and V_{M_m}) may not be crossed regardless of further information concerning higher moments, except that V_{P_p} and V_{M_m} may cross the hard bounds due to the point process approximations and the Bernoulli loop ramifications. The term ***hard bounds*** will be used to describe the solution values that result (recognize these as the *potential* values).

(2) An upper bound on program execution time may be obtained by assuming all computation state sojourn times and all connection times to be exponentially distributed with mean equal to the given first moments. This is believed to give an approximate upper bound on execution times, waiting times, V_{P_p} and V_{M_m} while it gives lower bounds on PE and GRM utilizations. These bounds are approximate in that computing the bounds requires ***the use of the model calculations which are approximate.*** Assuming exponential CDF's leads to overestimation of queueing times (they generally increase with the randomness of the arrival stream) and hence an overestimation of reference state sojourn times. This bounding relies on the assumption that program/processor timing is less randomly distributed than an exponential, this may not be the case in a general requestor/resource application so these bounds may be inappropriate in a more general environment. In practice the exponential based bounds have in fact held. These are not hard bounds in that they may not always hold due to the approximations involved. ***Term these bounds the exponential based bounds.***

(3) Another bound along the same lines may be developed. This bound is a soft lower bound on sojourn and waiting times, state transition times, coefficients of variation, and a soft upper bound on PE and GRM utilizations. In a similar manner to the exponential based bounds, assume all sojourn and connection times to be deterministic with values based on their means. Assuming deterministic timing induces discrete request stream interarrival timing and consequently underestimates waiting times. Hence reference state sojourn times will be lower bounded, utilizations will be upper bounded. Notice that system configurations with constant sojourn and

connection times will meet this set of bounds. These are termed *deterministic based bounds.*

In computing the bounds shown in chapter 6 the same solution technique is used as in chapter 4 with Q_{pm} *held at the value assumed during the model solution* obtained when using all the moments. In practice this is not be possible if only first moments are available, Q_{pm} would more realistically take on the value computed during the bound solution. Computing bounds increases total system analysis complexity, but the results of the analysis are more useful. The bounds form an approximate prediction interval in which the system behavior will be contained. In practice the bounds have not been trivial, i.e., the interval is not too large to be useful.

5.2. General Relationships

Consider attempting to set all waiting times to zero. An interesting issue regarding resource contention arises: some programs can not execute without some form of queueing of requests, contention must exist. Consider the fact that $\rho_m \leq 1 \ \forall m$. Then

$$\lambda'_m \overline{X}_m = \rho_m \leq 1 \qquad \forall m$$

From (3.38)

$$\overline{X}_m = \frac{1}{\lambda'_m} \sum_p \lambda_{pm} \overline{Y}'_{pm}$$

Then

$$\sum_p \lambda_{pm} \overline{Y}'_{pm} = \rho_m \leq 1 \quad \forall m. \qquad (5.1\bullet)$$

Since $\overline{Y}'_{pm}$ is given and does not depend on λ_{pm}, this bounds λ_{pm}; hence programs may not execute without some form of queueing effects that reduce λ_{pm} so that (5.1) is satisfied. Since there are M inequalities and $P \times M$ unknowns (λ_{pm}) there exist multiple solutions to these bounds.

The existence of multiple solutions to the relation

$$\sum_p \lambda^*_{pm} \overline{Y}'_{pm} = 1 \qquad \forall m$$

implies that a simple bound for λ_{pm} (i.e., $\lambda_{pm} \leq \lambda^*_{pm}$) may not be determined from (5.1). This results from the reduction in dimension seen in the relationships. $P \times M$ unknowns (λ_{pm}) are mapped into an M-vector (ρ_m). Simple bounds do exist if $\eta'_{pm} = 1$,

$\eta'_{jm} = 0, j \neq p$ (disjoint reference patterns).

The multitude of in solutions to (5.1) coincides with the existence of many queueing disciplines that will cause (5.1) to be satisfied. For example, various queue service disciplines may be derived that allow preferential treatment for certain PE's and discriminated treatment for others. Preferred PE's may access resources rapidly while others are slowed down in an effort to keep the utilization bound satisfied.

These bounds on λ_{pm} limit system operation; since the hard bounds described above may violate these utilization bounds, they may be spurious.

A relationship among PE and GRM utilizations exists and is invariant under queueing discipline changes: let $P(t)$ be the number of requests in the entire system at time t (a request may be "in" a PE if the PE is in a computation state), then

$$\begin{aligned} P(t) &= \sum_p 1_{C_p(t)} + \sum_m \tilde{N}_m(t) + \sum_m \tilde{B}_m(t) \\ &= \sum_p 1_{C_p(t)} + \sum_m \tilde{N}_m(t) + \tilde{B}(t) \end{aligned}$$

Where $C_p(t)$ is the event that PE p is in a computation state at time t, i.e., $C_p(t) = \{\tilde{Z}_p(t) \in \Omega_{C_p}\}$. The single request per PE case is equivalent to a constant number of requests in the system, $P(t) = P,\ t \geq 0$. If all processes in the system are ergodic then limits and expectations may be assessed to get

$$P = \sum_p \phi_p + \sum_m \overline{N}_m + \overline{B}$$

Potentials may be evaluated also ($\overline{N}_m = 0$ for potentials):

$$P = \sum_p \hat{\phi}_p + \sum_m \hat{\rho}_m$$

Then an interesting expression for the sum of mean queue lengths may be written:

$$\sum_m \overline{N}_m = \sum_p (\hat{\phi}_p - \phi_p) + \sum_m (\hat{\rho}_m - \rho_m) \tag{5.2}$$

For a symmetric situation ($P = M$, all programs are identical and reference patterns are uniform) this becomes:

$$\begin{aligned} P\,\overline{N}_m &= P(\hat{\phi}_p - \phi_p) + P(\hat{\rho}_m - \rho_m) \\ \overline{N}_m &= \hat{\phi}_p - \phi_p + \hat{\rho}_m - \rho_m \end{aligned}$$

Consider the calculation of $\overline{W}_m$ (an arbitrary request mean waiting time similar to $\overline{\overline{W}}_m(p)$ but with PE p included) with Little's formula applied to an arbitrary arriving request (i.e., without regard to type), then:

$$\overline{N}_m = \lambda_m^{'} \, \overline{W}_m = \lambda_m^{'} \left(\frac{1}{\lambda_m^{'}} \sum_p \lambda_{pm} \overline{W}_{pm} \right) = \sum_p \lambda_{pm} \, \overline{W}_{pm} \qquad (5.3\bullet)$$

Which relates mean *outside* observer time limiting (or Poisson examination time) queue lengths to the arrival point waiting times. $\lambda_{pm}\overline{W}_{pm}$ is the mean queue length for GRM m attributable to PE p. The relationship maps a matrix of $P \times M$ waiting time values into a vector of M queue length values. ***This reduction in dimension makes the inversion of $\overline{W}_m$ into $\overline{W}_{pm}$ indeterminable.*** It seems that an expression for $\overline{W}_{pm}$ must be derived for the asymmetric case.

Using (5.2):

$$\sum_m \sum_p \lambda_{pm} \overline{W}_{pm} = \sum_p (\hat{\phi}_p - \phi_p) + \sum_m (\hat{\rho}_m - \rho_m)$$

This relates waiting times and rates to system utilizations. For the symmetric case above

$$MP\lambda_{pm} \overline{W}_{pm} = M\overline{N}_m$$

$$P\lambda_{pm} \overline{W}_{pm} = \hat{\phi}_p - \phi_p + \hat{\rho}_m - \rho_m$$

but $\overline{W}_{pm} = \overline{W}_m$ for the symmetric case so

$$P\lambda_{pm} \overline{W}_m = \hat{\phi}_p - \phi_p + \hat{\rho}_m - \rho_m$$

5.3. On the Sensitivity of Utilizations

It has been found that utilizations are typically robust performance measures. That is, they may be predicted with sufficient accuracy even when using gross simplifying assumptions. The reason for GRM utilization insensitivity as often employed in early models [Bha75, Hoo77, MuM82a, Rau79, SeD79, Str70] may be seen from (consider for simplicity in notation, the symmetric case, i.e., where $\rho_m = \rho$, $\overline{W}_{pm} = W$, $\overline{Y}_{pm}^{'} = Y$, $\overline{S}_{pl}^{'} = S$, $\eta_{pm}^{'} = 1/M$, $\lambda_m^{'} = \lambda^{'}$, $\phi_p = \phi$):

$$\rho = \lambda^{'} Y$$

$$\lambda^{'} = \frac{P}{M(S + Y + W)}$$

$$\rho = \frac{PY}{M(S + Y + W)} \qquad \phi = \frac{S}{S + Y + W}$$

Define the following *sensitivities:*

$$\chi_M = \textit{GRM utilization sensitivity to changes in W}$$

$$\chi_P = \textit{PE utilization sensitivity to changes in W}$$

Then

$$\chi_M = \left| \frac{\partial \rho}{\partial W} \right|_{\text{solution } W} = \left. \frac{PY}{M(S+Y+W)^2} \right|_{\text{solution } W}$$

$$\chi_P = \left| \frac{\partial \phi}{\partial W} \right|_{\text{solution } W} = \left. \frac{S}{M(S+Y+W)^2} \right|_{\text{solution } W}$$

These functions are plotted for some representative cases in figure 5.1. It has been assumed that S and Y are ***constant with respect to W*** since we are interested in small perturbations in W due to prediction error. These functions demonstrate that around the solution point for W (which is approximate because the model solution has some error associated with it, simulation could have been used but evaluation would have been more expensive) utilizations are not very sensitive for those cases considered previously [Hoo77, MuM82a] where the mean computation state sojourn time has often been ≥ 1. As expected, the greatest sensitivity will be seen when $\phi_p << 1$, or $\overline{S}'_{p1} << \overline{Y}_{pm}$. Even in cases where $\overline{S}'_{p1} < \overline{Y}'_{pm}$, the sensitivity is relatively small. Hence the early models are able to achieve accurate results for utilizations even when simplifying assumptions are employed.

5.4. Derivatives

Interesting relationships among derivatives are shown below. Consider the fully symmetric case.

$$\frac{\partial \phi}{\partial Y} = - \frac{S\left(1 + \frac{\partial W}{\partial Y}\right)}{(S+Y+W)^2}$$

$$\frac{\partial \phi}{\partial S} = \frac{Y + W - S\frac{\partial W}{\partial S}}{(S+Y+W)^2}$$

$$\frac{\partial \rho}{\partial Y} = P\left(\frac{S + W - Y\frac{\partial W}{\partial Y}}{(S+Y+W)^2} \right)$$

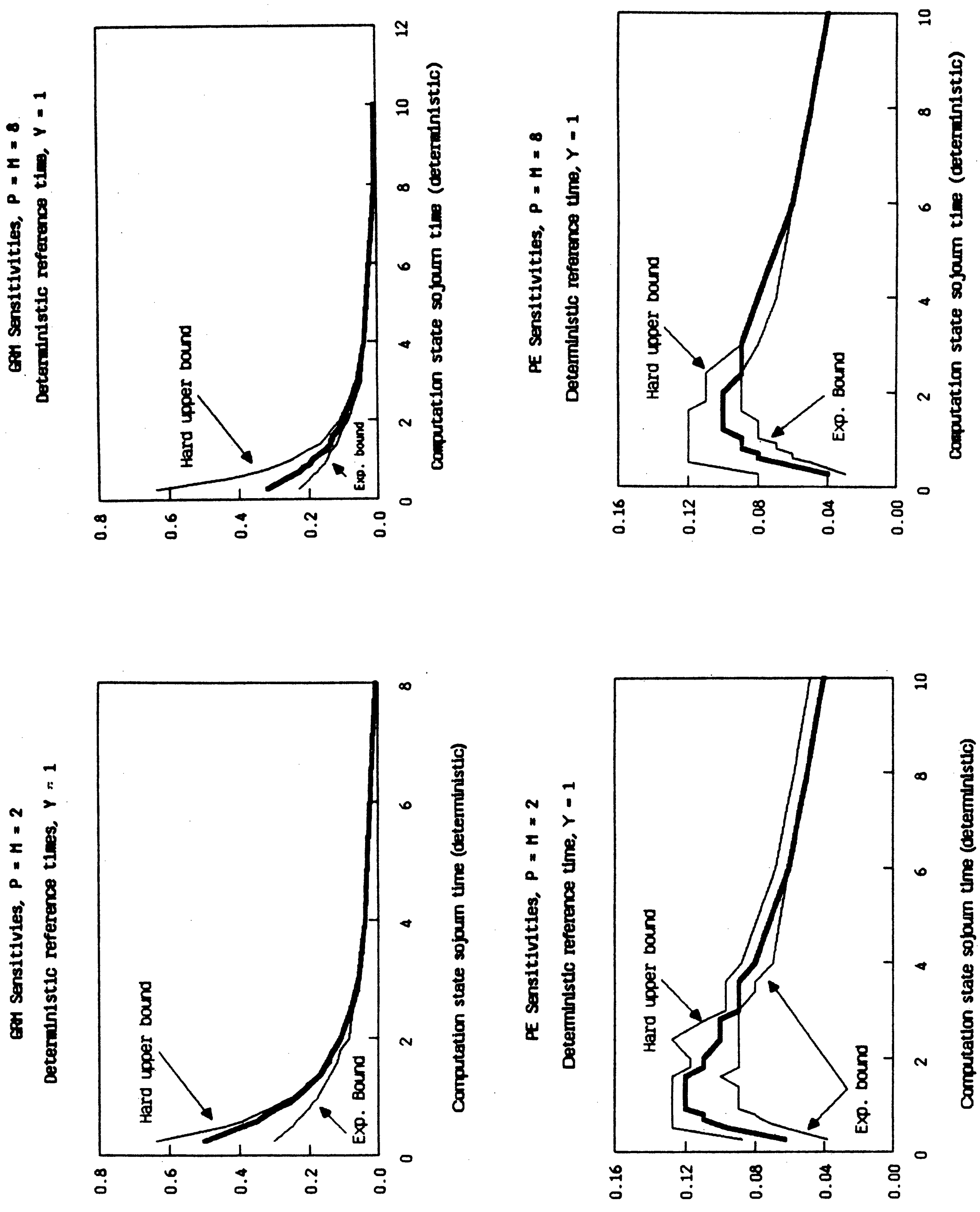


Figure 5.1 Sensitivity curves.

$$\frac{\partial \rho}{\partial S} = - \frac{PY\left(1 + \frac{\partial W}{\partial S}\right)}{M(S + Y + W)^2}$$

From which the following relationships become evident (solve for $\partial W/\partial S$ and set quantities equal):

$$\frac{\partial \phi}{\partial S} = \frac{MS}{PY}\left(\frac{\partial \rho}{\partial S}\right) + \frac{1}{S + Y + W}$$

Then

$$\lambda^{o} = \frac{\partial \phi}{\partial S} - \frac{MS}{PY}\left(\frac{\partial \rho}{\partial S}\right)$$

Which ties together PE request emission rates and system utilization derivatives. The following relationship connects utilization derivatives (with respect to mean connection times):

$$\frac{\partial \rho}{\partial Y} = \frac{P(S - Y + W)}{M(S + Y + W)^2} - \frac{PY}{MS}\left(\frac{\partial \phi}{\partial Y}\right)$$

Next consider formulating differential equations to find mean waiting times.

Use the chain rule to find $\partial W/\partial S$ and $\partial W/\partial Y$ in that S and Y form the *given* quantities in the simplified case described here. The equations would be solved for $W(S,Y)$. The approximation that W is only first moment dependent on S and Y will be used throughout. This is certainly not true in general, the applicability of the differential equation approach depends on the amount of accuracy desired. This is an experimental approach that has not been tested. It is believed that more work is required here in testing the validity of, and solving these equations. It is believed that physical analysis could be done to obtain approximations for the fundamental functions required by the analysis:

$$\frac{\partial W}{\partial S} = \frac{\partial W}{\partial \rho}\,\frac{\partial \rho}{\partial S}$$

Where $\partial W/\partial \rho$ is to be obtained from physical analysis of the queueing station, it would depend on results from queueing theory. Then:

$$\frac{\partial W}{\partial S} = -\frac{PY\left(\frac{\partial W}{\partial \rho}\right)}{M(S+Y+W)^2 + PY\left(\frac{\partial W}{\partial \rho}\right)}$$

Similarly, writing an expression for $\partial W/\partial Y$:

$$\frac{\partial W}{\partial Y} = \frac{\partial W}{\partial \rho}\,\frac{\partial \rho}{\partial Y}$$

The following is obtained:

$$\frac{\partial W}{\partial Y} = \frac{P(S+W)\left(\frac{\partial W}{\partial \rho}\right)}{M(S+Y+W)^2 + PY\left(\frac{\partial W}{\partial \rho}\right)}$$

Notice that $\partial W/\partial S$ and $\partial W/\partial Y$ form partial differential equations in S and Y. No further development of the differential equation approach has been pursued.

CHAPTER 6

SIMULATION EXPERIMENTS

This chapter describes experimental data obtained in an effort to verify and validate the SMP model and the calculations shown in the proceeding chapters. In particular, two classes of simulation experiments have been performed: program execution experiments intended to validate the SMP model assumptions; and synthetic experiments intended to verify the calculations.

In the first class, two program execution simulations have been developed: a list merge experiment (one phase of a merge sort program); and an elementary database processing program.

The second class consists of experiments of a synthetic nature. These are controlled experiments in which calculation error is examined. Specifically, waiting time calculations have been tested with the synthetic experiments.

The SMP model has been found to be generally accurate. In the program execution level experiments, execution times are typically predicted within about 5% of their simulated value. GRM utilizations may display up to about 15% error (this error is seen in the database experiment where the system is predominantly disk I/O bound, it is a difficult prediction problem because of the disk bottleneck, $\phi_p < 0.05$ in these experiments). Waiting time error may approach 30% when considering total GRM reference time (waiting and connection time). The waiting time prediction problem is comparable to the prediction of mean waiting times in finite customer, multi-class, queueing networks.

The first synthetic simulation example concerns asymmetric ARP simulation data. The accuracy of the various approximations is examined in a highly unpredictable and asymmetric situation.

The second synthetic simulation example concerns the approximation that $\overline{W}_{psm} \simeq \overline{W}_{pm}$. In particular, two much different reference states have been placed in PE descriptions in an attempt to obtain a measurable -- using cycle time data -- difference in waiting times for the two states. The results of this experiment support the approximation that $\overline{W}_{pm}$ is almost $\overline{W}_{psm}$ when loop cycle times, utilizations, rates, and approximate coefficients of variation are used as performance indicators (i.e. when $\overline{W}_{psm}$ is not required with a high degree of accuracy).

A comparison with a previous synchronous system model is presented. This is done to see if the waiting time calculations break down in a discrete time, synchronous system activity environment.

6.1. Program Execution Experiments

In both of these examples, it will be assumed that instructions and local data are stored in local memory so that GRM references are global data or resource references. This form of system operation allows PE activity to be represented by program flowcharts (once compiled into a PE state diagram) directly, program execution time is of primary concern.

The simulators are in themselves operational SIMSCRIPT II.5 programs into which timing specifications have been inserted. These timing specifications are believed to represent object code execution timing and associated connection times. Although exact timing emulation of an existent system has not been stressed, the values chosen are believed to be accurate within the objectives of the experiment. High precision in emulating specific system timing specifications is not required to obtain a valuable experiment.

6.1.1. The Database Example

Consider a simple airline reservation system composed of terminals, PE's and a memory system as shown in figure 6.1. The sub-system composed of PE's and the database storage system is assumed to be devoted to the storage and processing of airline reservations; that is, it is a dedicated system. Terminal users converse with the system regarding reservations for customers and manipulate the database. The database in this example consists of records of information regarding the occupants of seats on flights. The records appear as in figure 6.2.

In this example, terminal users generate inquiries into flight occupation status, reserve seats for customers on given flight numbers, etc. Five simple commands have been implemented in the simulator:

Add a customer to a given flight number.

Remove a customer from a given flight number.

List information on all customers on a given flight number.

Retrieve information on the customer in a given seat on a given flight number.

Count the number of passengers on a given flight number.

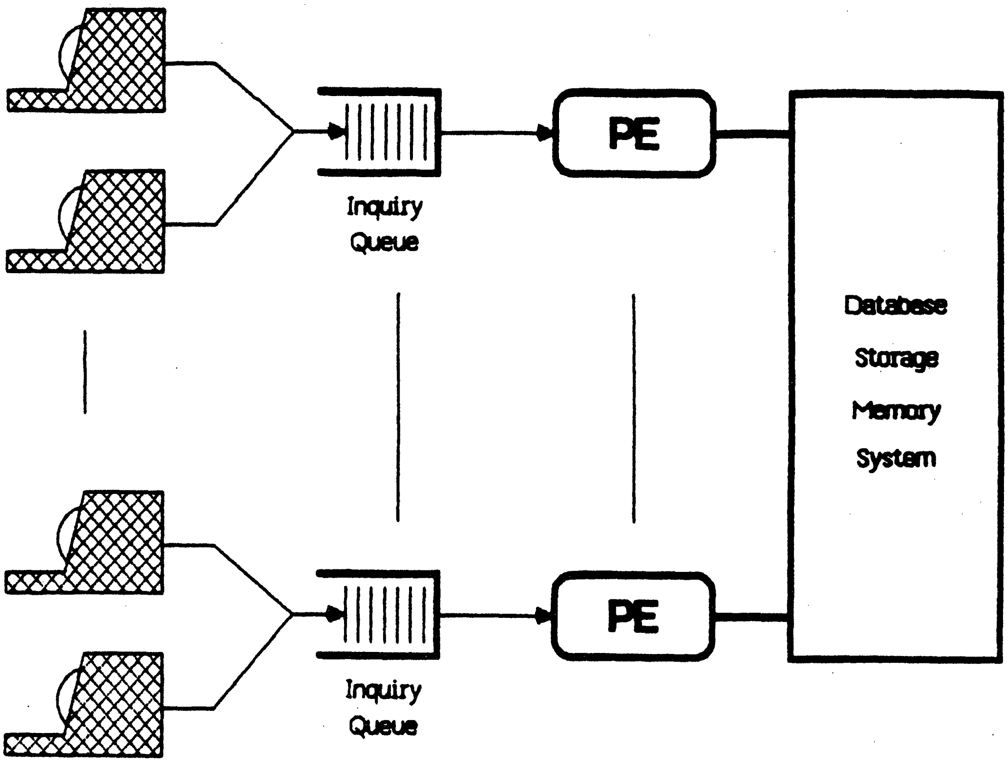


Figure 6.1 Airline reservation system.

Flight #	Seat #	Customer Information

Figure 6.2 Database record.

Performance measures of interest include the inquiry time statistics (for example, the mean response time) and again, system element utilizations. The inquiry time is believed to be most important in that if terminal user inquires are queued (as in figure 6.1) it is informative to predict the queueing time statistics. The first two moments of inquiry processing times are required if an M/G/1 inquiry queue model is to be used.

Although simplistic it will be assumed that inquiries are processed sequentially, each PE accepts an inquiry the processes it to its completion before accepting the next inquiry from the inquiry queue. To do otherwise would violate the single request per PE assumption. Even though the single request per PE assumption is restrictive in this case (multiprogramming of PE's would be used in practice owing to the slow nature of the disk units on which the database is stored) the experiment is worthwhile as a test of the SMP model; future work might be directed toward relieving the monoprogrammed assumption. See chapter 8 for the difficulties involved. Multiple request emission modeling is an obvious SMP model enhancement for consideration.

6.1.1.1. Implementation

Consider the physical storage of the database in the memory system. Lists of pointers will be kept in RAM modules (a subset of the set of GRM's are RAM modules), and the database of seat occupation records will be kept in disk modules which are the remainder of the GRM's. A request queue is present in front of each of the resource modules.

A two level directory of pointers is organized as in figure 6.3. The *directory elements* are stored in an interleaved manner across RAM modules. Customer records are uniformly distributed over all disk units. That is, a seat record may be stored anywhere on the disk system, they are not clustered together into flight numbers. This represents the randomness or disorder evident in systems where little is known about flight seats and/or no disk repacking is done and fragmentation reaches a steady-state. More sophisticated disk allocation schemes would make the disk analysis phase (covered below) more complex without sufficiently increasing validity of the example as an SMP model test. Consider next the implementation of the five database commands: add; remove; list; retrieve; and count.

6.1.1.2. The ADD Command

This command places passenger data into a given flight and seat number database record. The customer's record (consisting of a name, an address, etc., assumed to be 256 bytes of information) is placed in a disk record pointed to by the two level directory (accession list). Note that since disk sectors may be larger than 256 bytes, there may be several customer records in one sector. If an attempt is made to place a customer into a used seat, it is ignored (that is, the error causes no action to be taken) although it does take some simulated time. A flowchart/PE state diagram of operation is shown in figure 6.4. Notice that the circuit switched property is used to modify the "seat in use" field when a customer is added (i.e., semaphores are handled).

6.1.1.3. The REMOVE Command

This command marks a given seat number, on a given flight number, as not in use so that it may be allocated in the future. Note that no disk I/O is required, only pointer lists in RAM are modified. An appropriate flowchart/PE state diagram is shown in figures 6.5. If an attempt is made to remove a customer from an unoccupied seat, no action is taken but simulated time elapses.

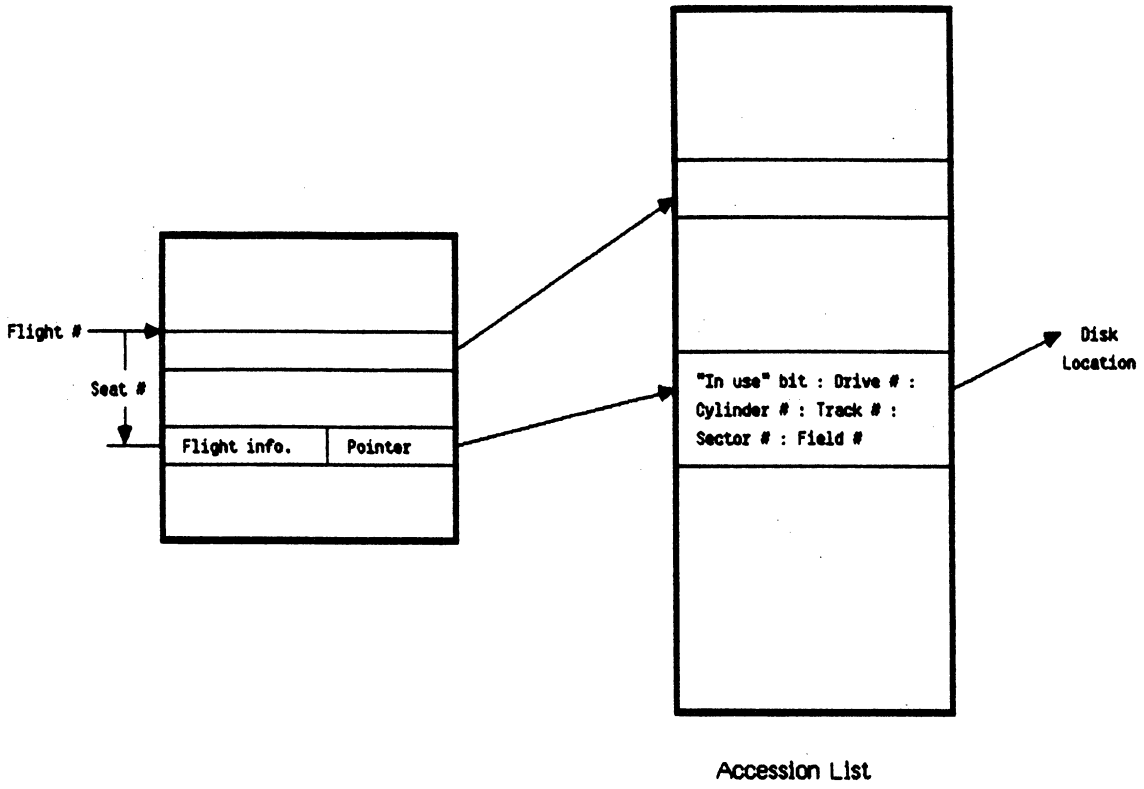
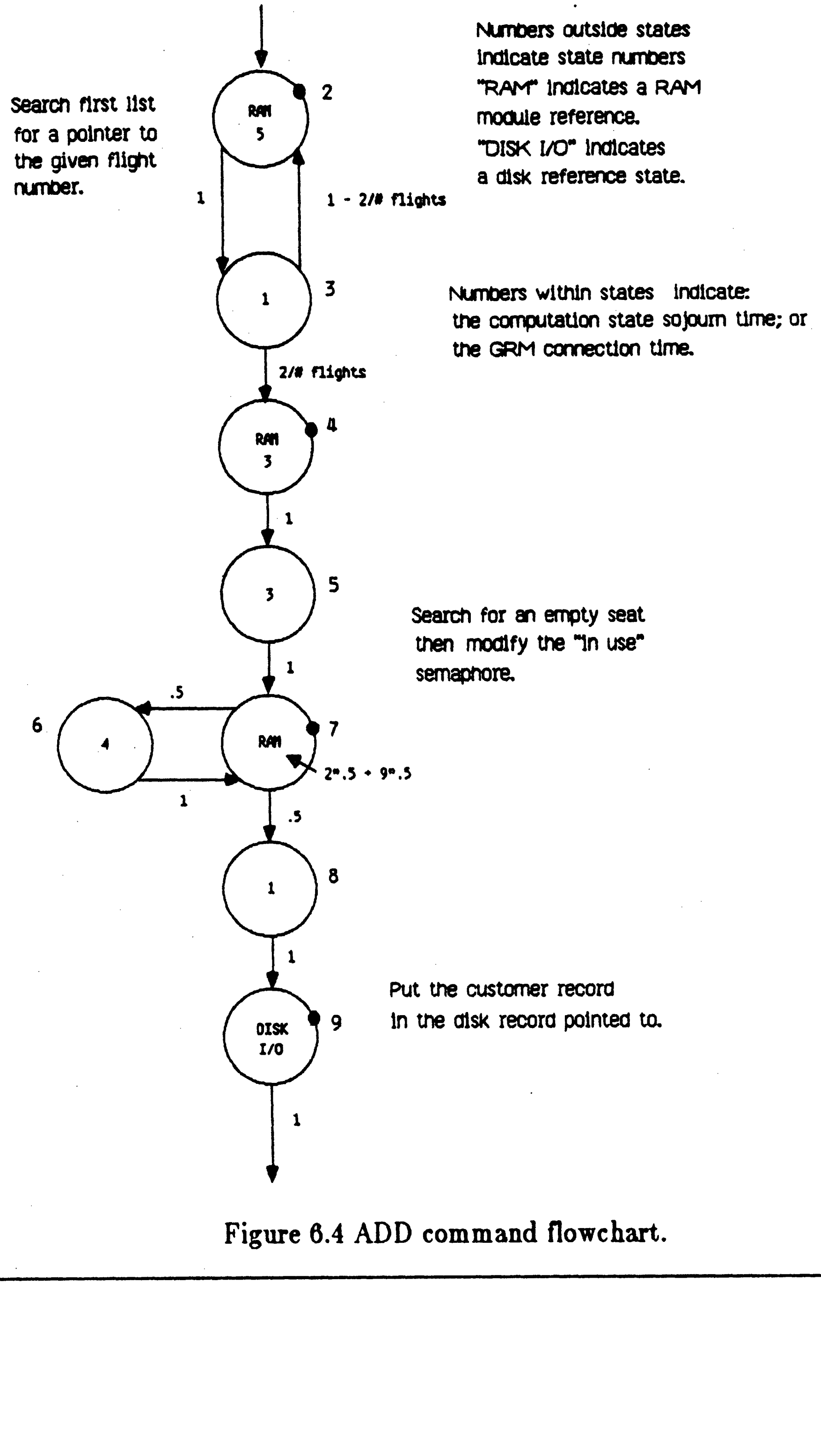


Figure 6.3 Two levels of pointers.



Search first list for a pointer to the given flight number.

Numbers outside states indicate state numbers "RAM" indicates a RAM module reference. "DISK I/O" indicates a disk reference state.

Numbers within states indicate: the computation state sojourn time; or the GRM connection time.

Search for an empty seat then modify the "in use" semaphore.

Put the customer record in the disk record pointed to.

Figure 6.4 ADD command flowchart.

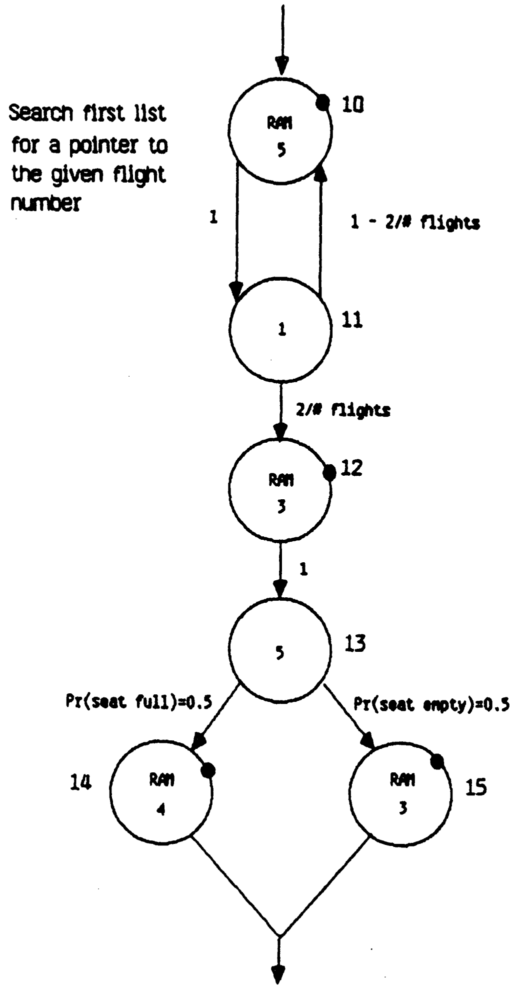


Figure 6.5 REMOVE command flowchart.

6.1.1.4. The LIST Command

This command retrieves all records of all passengers on a given flight number. It does this by sequentially accessing all records in use for the required flight number. As such, many disk accessed are performed during this command's execution. A flow chart/PE state diagram is shown in figure 6.6

6.1.1.5. The RETRIEVE Command

This command retrieves the information on a passenger in a given flight, seat number. This entails accessing the appropriate disk record. A flowchart/PE state

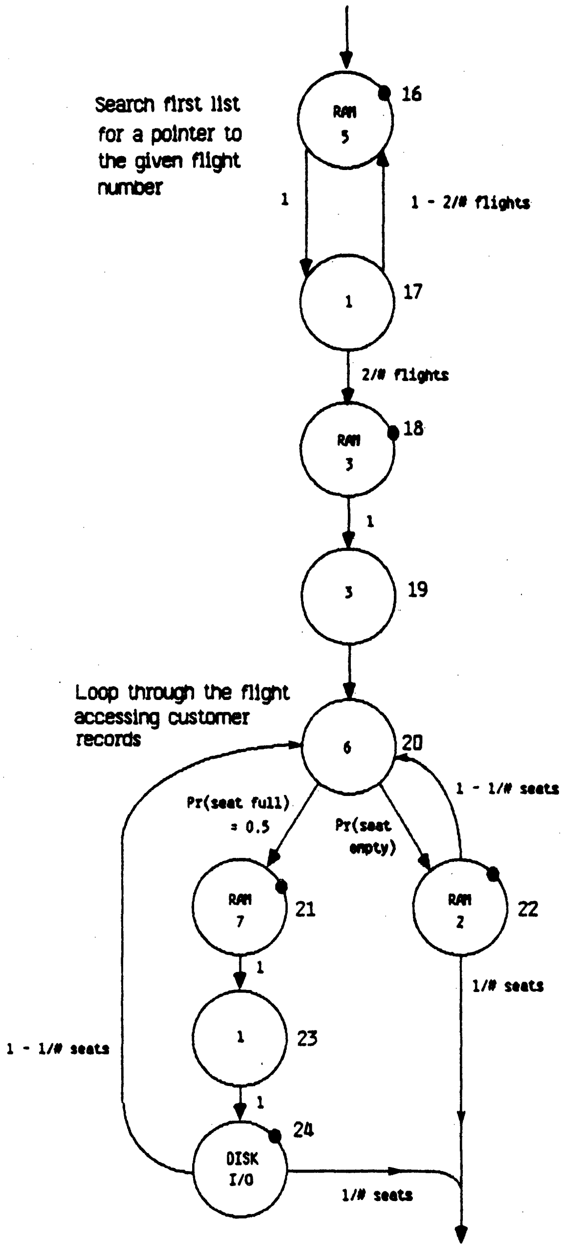


Figure 6.6 LIST command flowchart.

diagram is shown in figure 6.7. If a retrieval on an unoccupied seat is requested, no action takes place other than checking the accession list.

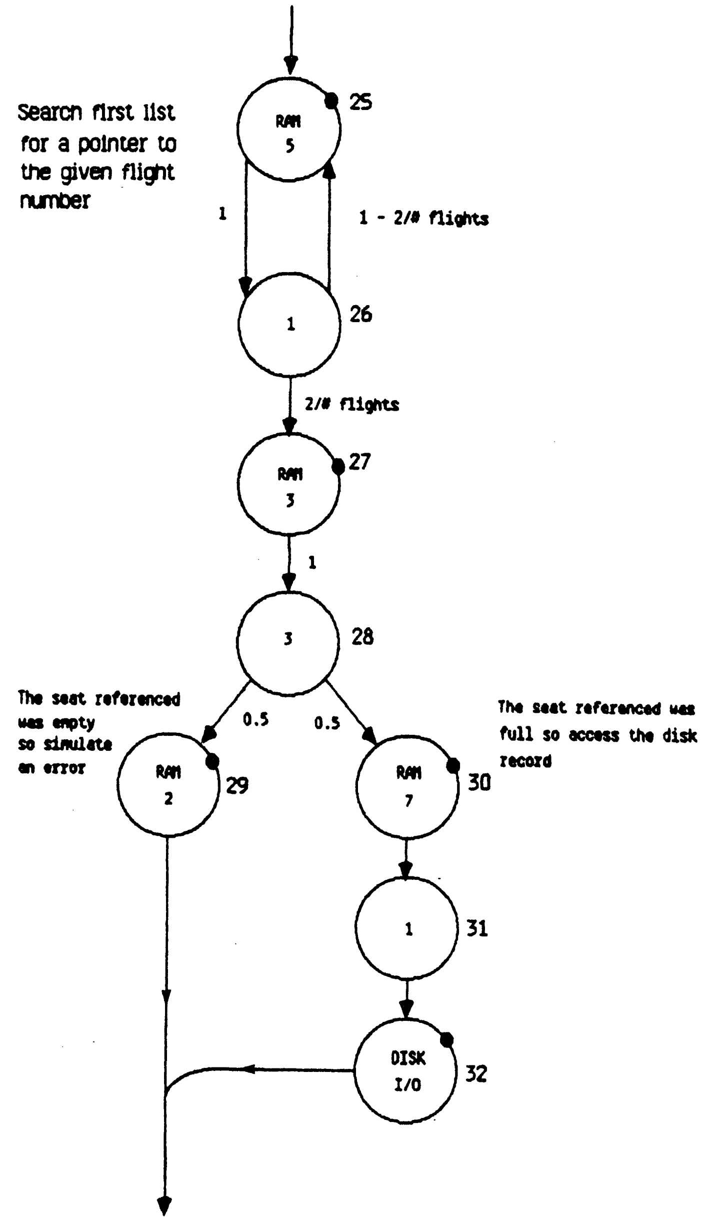


Figure 6.7 RETRIEVE command flowchart.

6.1.1.6. The COUNT Command

This command counts the number of passengers presently scheduled on the referenced flight number. Notice that no disk accesses are made, only pointer lists are referenced. See figure 6.8.

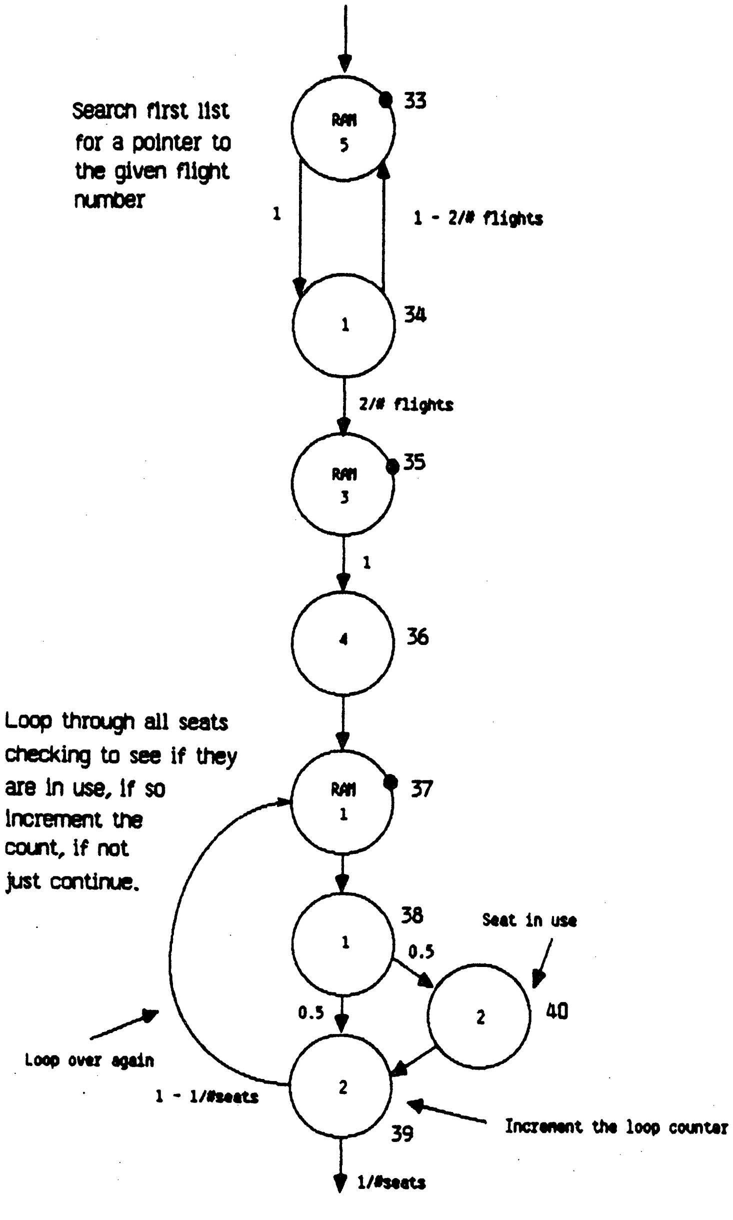


Figure 6.8 COUNT command flowchart.

6.1.1.7. The Simulator

Commands submitted to processors arrive as random streams. (There are P processors executing their database programs independently.) This randomness represents the independence between user inquiries that might arise if many terminals are connected to each PE inquiry queue so that successive inquiries originate from different terminals.

Hence we assume that command frequencies appropriately depict inquiry streams. Likewise flight and seat numbers are randomly chosen in the simulator when commands are initiated. This is sometimes inaccurate in that a REMOVE command would not normally be attempted on an empty seat, but in the simulator many may occur. Although not very realistic, it simplifies simulation and as long as the representation of simulator behavior is duplicated during model parameter extraction the experiment is still valid. (The simulator source code was examined to extract the model parameters.) The commands are tied together as in figure 6.9 where the top state's sojourn time is chosen to represent the amount of time that it takes to obtain the next inquiry from the inquiry queue. It has been assumed that there is always an inquiry ready to be processed. If this were not the case, then a random sojourn time would be used for the top state, again this is just another possibility that the SMP model parameters would reflect.

Statistics have been gathered on GRM utilizations, waiting times, GRM arrival stream coefficients of variation, GRM mean queue lengths (they will not be shown due to their direct relationship to waiting times, see equation (5.3)), and the mean command execution time.

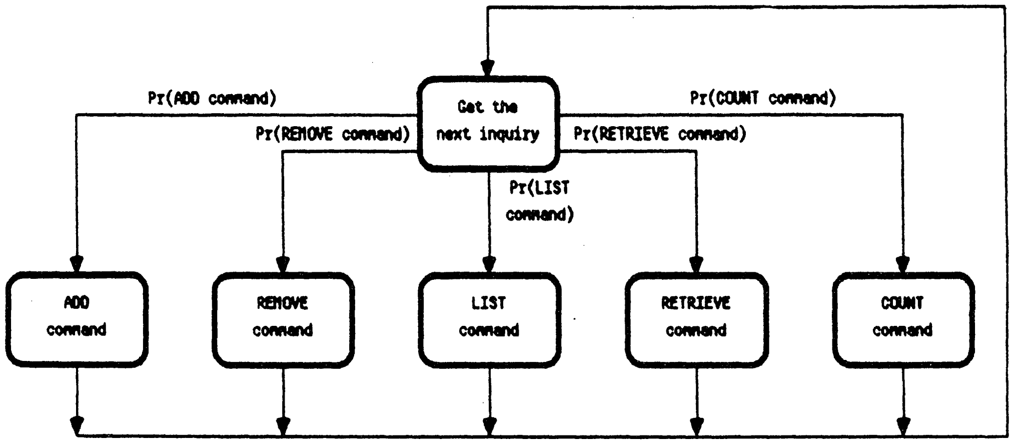


Figure 6.9 Connecting the commands together.

6.1.1.8. Model Parameter Extraction

Computation state sojourn and RAM connection times have been deduced from the simulator code (or flowchart). Disk access time calculations are required to obtain the first three moments of disk access (connection) times. The calculations follow:

Let there be C cylinders on the disk drive under consideration. Also use the following notation:

$\tilde{A}$ = disk access time =
time required to move the disk head, wait for
the required sector to move under the head,
and read the sector

$\tilde{M}$ = time required to move the head from its
present position to the next required cylinder

$\tilde{L}$ = rotational latency seen once the required
cylinder has been arrived at

$\tilde{T}$ = sector transfer time

Then $\tilde{A} = \tilde{M} + \tilde{L} + \tilde{T}$. Disk module connection time is $\tilde{A}$ so disk I/O states have connection time moments $\overline{A^{1,2,3}}$. Since $\tilde{M}$, $\tilde{L}$ and $\tilde{T}$ are independent:

$$\bar{A} = \bar{M} + \bar{L} + \bar{T}$$

$$\overline{A^2} = \overline{M^2} + \overline{L^2} + \overline{T^2} + 2(\bar{L}\bar{M}+\bar{T}\bar{M}+\bar{L}\bar{T})$$

$$\overline{A^3} = \overline{M^3} + \overline{L^3} + \overline{T^3} + 3[(\bar{L}+\bar{T})\overline{M^2} + (\bar{M}+\bar{T})\overline{L^2} + (\bar{M}+\bar{L})\overline{T^2}] + 6\bar{M}\bar{L}\bar{T}$$

The transfer time random variable $\tilde{T}$ is constant so it is obtainable from disk drive specifications.

The rotational latency is uniformly distributed from 0 to a maximum value of Γ in the environment considered here, assuming that there is negligible dependency of the rotational position on PE inter-disk I/O computation time. Then $\bar{L} = \Gamma/2$; $\overline{L^2} = \Gamma^2/3$; $\overline{L^3} = \Gamma^3/4$.

Consider now the disk motion time moments. Let $\tilde{P}$ be the cylinder where the disk head rests at the initiation of the disk access $\tilde{P} \sim U(1,C)$ (discrete) because of the random scattering of data records over all disk structures. Let $\tilde{N}$ be the cylinder to be stepped to $\tilde{N} \sim U(1,C)$ (discrete). Then the distance to be moved is $|\tilde{P}-\tilde{N}|$. The head stepping time function is shown in figure 6.10 where stepping time has been linearly interpolated.

Let q_j be the probability that j cylinders must be traversed, then:

$$q_j = Pr(|\tilde{P}-\tilde{N}| = j)$$

$$= \begin{cases} \frac{1}{C} & j = 0 \\ \sum_{k=1}^{j} Pr(\tilde{N} = j+k)\, Pr(\tilde{P} = k) & \\ + \sum_{k=j+1}^{C-j} (Pr(\tilde{N} = j+k) + Pr(\tilde{N} = k-j))\, Pr(\tilde{P} = k) & 1 \le j \le C-1 \\ + \sum_{k=C-j+1}^{C} Pr(\tilde{N} = k-j)\, Pr(\tilde{P} = k) & \end{cases}$$

But $Pr(\tilde{N} = k) = \frac{1}{C}$, $k = 1, \cdots, C$, $Pr(\tilde{P} = k) = \frac{1}{C}$, $k = 1, \cdots, C$. So a closed form might be obtained.[1]

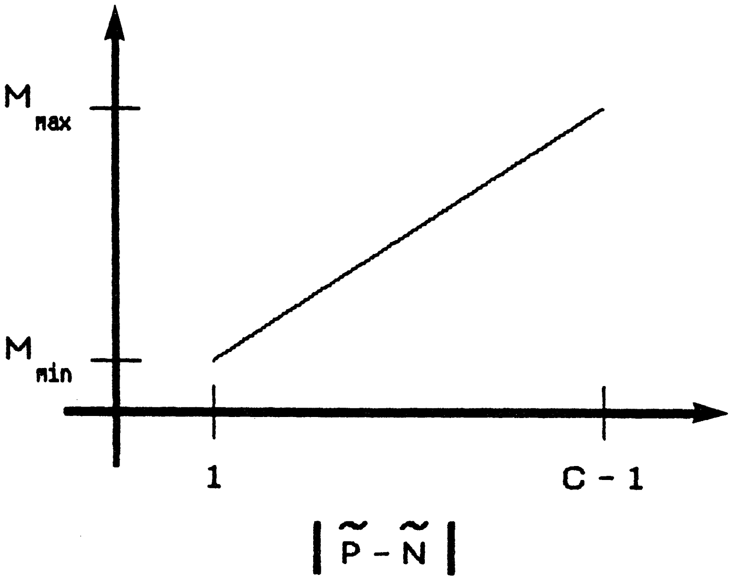


Figure 6.10 Disk head stepping time function.

[1] It has not been pursued, the mean has been found to be about C / 3 for the number of cylinders stepped.

From figure 6.10, the motion time is related to the distance to be stepped as:

$$\tilde{M} = \begin{cases} m(|\tilde{P}-\tilde{N}|-1) + M_{\min} & |\tilde{P}-\tilde{N}| > 0 \\ 0 & |\tilde{P}-\tilde{N}| = 0 \end{cases}$$

Where m is the slope of the stepping time curve:

$$m = \frac{M_{\max} - M_{\min}}{C - 2}$$

then

$$\overline{M} = 0 \cdot q_o + \sum_{k=1}^{C-1} (m(k-1) + M_{\min}) q_k$$

$$\overline{M^2} = \sum_{k=1}^{C-1} (m(k-1) + M_{\min})^2 q_k$$

$$\overline{M^3} = \sum_{k=1}^{C-1} (m(k-1) + M_{\min})^3 q_k$$

Which completes a randomly accessed disk access time description.

Connection times for the drives in the analysis are taken from the above calculations where parameters $M_{\min}$, $M_{\max}$, and C are taken from simulator parameters. That is, the physical parameters are the same for both the simulation and the analysis. The simulator remembers $\tilde{P}$ from the last disk access and uses $\tilde{N}$ from the accession list so head motion simulation is realistic. The simulator maps $|\tilde{P} - \tilde{N}|$ into $\tilde{M}$ using the linear interpolation above. The simulator uses a uniform random variable to simulate rotational latency $\tilde{L}$. At the time of simulation $C - 1$ was inadvertently used in the slope curve. Hence the denominator $C - 1$ was also used in the analysis.

Transition probabilities have been deduced intuitively from examining the program segment flowcharts. The Bernoulli loop has been extensively employed. Command probabilities have been chosen to induce a steady-state add/delete situation:

$Pr(ADD\ command) = Pr(REMOVE\ command) = .35,$
$Pr(RETRIEVE\ command) = .1$
$Pr(LIST\ command) = .1$
$Pr(COUNT\ command) = .1$

Assuming that all flights are initially half full (the simulator starts this way), they will stay approximately half full because $Pr(ADD) = Pr(REMOVE)$, furthermore the simulator initializes seats to be full/empty in an alternating pattern. These startup characteristics and command fractions were chosen so that the simulator reaches a steady-state in reasonable time. Other more sophisticated procedures could have been developed but simulation cost would have increased without significantly increasing the validity of the example.

6.1.1.9. Database Results

Four separate experiments have been performed. Complexity in simulating concurrent PE activity makes simulation expensive. This is an example where the analytic approach is comparatively economical, about a factor of 15 to 20 has been seen in the analytic verses single simulation cost, including the computation of the three bounding solutions described in chapter 5. Each simulation has been run three times with different random number generators to obtain reasonable accuracy in results, so the speed factor may actually be 45-60. Cost prohibits the compilation of confidence intervals. Each PE executes 2000 commands in a given experiment.

The four experiments are characterized as follows:

Experiment 1 - 2 PE's, 2 disk units, 2 global memory modules.

100 cylinders per disk, $M_{\min} = 500$, $M_{\max} = 2000$.

60 flights, 100 seats per flight.

Experiments 2 - the same as experiment 1 except with 3 PE's.

Experiments 3 - 2 PE's, 3 disks, 1 global memory module.

67 cylinders per disk (to maintain the same total disk storage).

$M_{\min} = 500$, $M_{\max} = 1333$.

60 flights, 100 seats per flight.

Experiment 4 - the same as experiment 3 except with 3 PE's.

The results are summerized on the following pages. Again, any parameters not discussed have been assumed in the simulator to represent "object code" execution time.

Experiment #1, 2 processors, 2 disks, 2 RAM modules								
Mean Command Execution Time	ρ_{DISK}	ρ_{RAM}	$\overline{X}_{DISK}$	$\overline{W}_{DISK}$	$\overline{W}_{RAM}$	$V_{M_{DISK}}$	$V_{M_{RAM}}$	
14462	0.714	0.016	1773	626	3.54	0.872	2.56	Simulation, 3 runs
15146	0.633	0.014	1774	946	0.083	1.11	1.82	G/G/1 solution
17754	0.540	0.012	1774	1428	0.137	1.16	2.09	Expon. Bound, G/G/1
14712	0.650	0.015	1774	866	0.082	1.11	1.77	Determ. Bound, G/G/1
12588	0.7611	0.017	1774	473	0.021	1.11	1.83	Case 1,2 SMP soln.
14678	0.6527	0.0145	1774	860	0.037	1.18	2.21	Exp. Bound, Case 1,2 SMP
12343	0.7762	0.0173	1774	427	0.022	1.10	1.76	Det. Bound, Case 1,2 SMP
10034	0.95	0.020	1774	0	0	1.09	1.66	Hard bound

Experiment #2, 3 processors, 2 disks, 2 RAM modules								
Mean Command Execution Time	ρ_{DISK}	ρ_{RAM}	$\overline{X}_{DISK}$	$\overline{W}_{DISK}$	$\overline{W}_{RAM}$	$V_{M_{DISK}}$	$V_{M_{RAM}}$	
20406	0.812	0.017	1775	1424	4.65	0.879	2.66	Simulation, 3 runs
20048	0.720	0.016	1774	1854	0.080	1.08	1.62	G/G/1 solution
22667	0.634	0.014	1774	2338	0.132	1.12	1.81	Expon. Bound, G/G/1
19790	0.726	0.016	1774	1806	0.080	1.08	1.59	Determ. Bound, G/G/1
15282	.9404	.021	1774	971	.0351	1.09	1.67	Case 1,2 SMP soln.
18532	.7755	.0173	1774	1573	.058	1.13	1.91	Exp. Bound, Case 1,2 SMP
14882	.9657	.0215	1774	897	.0359	1.08	1.62	Det. Bound, Case 1,2 SMP
10034	1.00	0.032	1774	0	0	1.06	1.46	Hard bound

$\phi_p < 0.05$ for the database tests.

Experiment #3, 2 processors, 3 disks, 1 RAM module								
Mean Command Execution Time	ρ_{DISK}	ρ_{RAM}	$\overline{X}_{DISK}$	$\overline{W}_{DISK}$	$\overline{W}_{RAM}$	$V_{M_{DISK}}$	$V_{M_{RAM}}$	
11463	0.527	0.040	1550	327	2.2	0.900	3.26	Simulation, 3 runs
11726	0.476	0.036	1551	533	0.3	1.072	2.31	G/G/1 solution
13460	0.415	0.032	1551	852	0.6	1.111	2.80	Expon. Bound, G/G/1
11462	0.487	0.037	1551	484	0.3	1.067	2.23	Determ. Bound, G/G/1
10303	0.5420	0.041	1551	272	.053	1.07	2.32	Case 1,2 soln.
11517	0.485	0.037	1551	496	.094	1.13	2.97	Exp. Bound, Case 1,2 SMP
10167	0.549	0.042	1551	247	.053	1.07	2.22	Det. Bound, Case 1,2 SMP
8831	0.632	0.048	1551	0	0	1.058	2.11	Hard bound

Experiment #4, 3 processors, 3 disks, 1 RAM module								
Mean Command Execution Time	ρ_{DISK}	ρ_{RAM}	$\overline{X}_{DISK}$	$\overline{W}_{DISK}$	$\overline{W}_{RAM}$	$V_{M_{DISK}}$	$V_{M_{RAM}}$	
14623	0.650	0.050	1550	737	3.4	0.930	3.3	Simulation, 3 runs
13700	0.611	0.047	1551	899	0.3	1.05	2.01	G/G/1 solution
15904	0.527	0.040	1551	1305	0.5	1.08	2.34	Expon. Bound, G/G/1
13375	0.626	0.048	1551	839	0.3	1.05	1.96	Determ. Bound, G/G/1
11724	0.715	0.055	1551	535	.09	1.06	2.03	Case 1,2 SMP soln.
13857	0.605	0.046	1551	929	.16	1.09	2.46	Exp. Bound, Case 1,2 SMP
11476	0.73	0.056	1551	489	.094	1.05	1.96	Det. Bound, Case 1,2 SMP
8830	0.949	0.073	1551	0	0	1.04	1.78	Hard bound

Note from the tables that the coefficient of variation for the request interarrival time at disk drives is generally overestimated. As a result of this, the G/G/1 based waiting time calculations overestimate queueing times (see the G/G/1 waiting time dependence on the interarrival time coefficient of variation (4.24)).

The source of the error seen in the prediction of the coefficient of variation may be from inaccuracies in (4.24) itself, the higher moment waiting time calculations (recall that the first *and* second moments of waiting times affect the coefficient of variation of the interarrival times, see equations (4.28 - 4.33) for details), the Bernoulli loop first moment matching technique (as described in section 3.2), and the superposition calculations. Point process superposition and separation calculation error is addressed shortly.

Consider experiment #1. The disk waiting times are overpredicted while the RAM waiting times are vastly underpredicted (note that $V_{M_{RAM}}$ is underpredicted). The results are not quite as bad as they appear, there is non-negligible additive statistical error on small waiting times that is attributable to the SIMSCRIPT II.5 simulation language. $\overline{W}_{RAM}$ is generally underpredicted, but not by as much as is indicated in the tables.

Consider next experiment #2. There are 3 PE's so GRM waiting times increase over experiment #1 as expected. Even though disk waiting times are overestimated here also, the mean command execution time is underpredicted using the G/G/1 results. This is attributable to the greater underprediction in $\overline{W}_{RAM}$ and the large fraction of RAM references compared to disk references: $\eta'_{RAM}=.4531$ while $\eta'_{DISK}=.0469$. These reference statistics are sufficient to ensure that the mean command execution time will be underpredicted even though disk waiting time is overestimated.

Also noted in the SMP to ARP reduction for the database program are the following statistics (all PE's behave similarly):

$$\overline{S}_{p1}' = 4.175, \quad \overline{S_{p1}^2}' = 185.47$$

$$\overline{Y}'_{p,RAM} = 4.1, \quad \overline{Y}'_{p,DISK} = 1774$$

$$\overline{Y^2_{p,RAM}}' = 20.3, \quad \overline{Y^2_{p,DISK}}' = 3.48\times 10^6$$

So the following coefficients of variation are deduced:

Coefficient of variation for
equivalent computation state 1 $= 3.3$

$$\text{Coefficient of variation for } \tilde{Y}'_{p,RAM} = 0.46$$
$$\text{Coefficient of variation for } \tilde{Y}'_{p,DISK} = 0.325$$

If these coefficients are accurate (the disk connection time coefficient of variation is quite accurate) then it may be seen that connection times are more deterministic than exponential CDF's describe, and the equivalent computation time is "more" random than an exponential CDF indicates. This coefficient though is subject to the same higher moment inaccuracies described in the V_{M_m} predictions.

The hard bounds apply in all cases *except* in the GRM arrival stream coefficients of variation. *This breakdown in the hard bounds leads to the belief that some of the most inaccurate model equations are the point process superposition and separation calculations.*

If the point process mechanics were more accurate, V_{M_m} for the hard bounds should be smaller than the simulated value. Inaccuracies may occur because of correlation between request interemission times not described by the ARP representation of PE activity. (Superposition theory is constrained by the assumption of a renewal resultant process which is not true in general.) This correlation is related to the history effects that were briefly described in section 3.3. Dependence between request emissions introduces correlation between ARP cycle times.

As is typically true of the various waiting time calculations, the G/G/1 approach yields greater waiting time results than the two SMP calculations. The SMP and G/G/1 results may form bounds on the actual waiting times, violations will be seen in the asymmetric example (the SMP calculations seem too inaccurate in vastly asymmetric situations). Notice from experiment #1 SMP results that when waiting times are underpredicted, GRM utilizations are overpredicted. Also note that the SMP results for waiting times are so small that the exponential bounds (which may overestimate waiting times) may fail to hold. This seems to indicate that there is too much error in the independent SMP waiting time calculations to be useful, especially in asymmetric situations where the results are expected to be worse due to the mean waiting time approximation (4.9 - 4.16).

The case 2 SMP results in this example are numerically equal to the case 1 results. The factors h_{pm}, f_{pm} are equivalent because $V_{M_m} > 1$.

Using the G/G/1 calculations has given execution time predictions within ±6% of the simulated value. Considering that $\phi_p < 0.05$ (the system is disk bound because of the single request per PE property) this error seems quite acceptable.

Regions of validity for the various approximations have not been quantitatively defined. The G/G/1 based solution is typically accurate enough to be used in general since the relative difference between the SMP and G/G/1 waiting time results may not be significant in those cases where the SMP results are best. That is, in the cases where the SMP waiting times are most accurate, the G/G/1 results are sufficiently accurate to be used. It seems that the G/G/1 based waiting time predictions are the most robust of the two presented in chapter 4.

6.1.2. List Merge Program

In this example P PE's merge independent preordered lists. Each PE merges two preordered lists contained in global memory to form a new ordered list kept in global memory. Since all PE's are acting on separate list elements, this is equivalent to the last full activity phase of a parallel merge sort, at the point where half of the PE's would subsequently become inactive.

Instructions and local data are stored in local memory. The lists to be merged are stored in global memory in an interleaved manner, see figure 6.11. The base address of each list (there are two source lists and one destination list for each PE) is mapped into GRM 1.

A flowchart of PE activity is shown in figure 6.12 while a state diagram is shown in figure 6.13. All GRM connections constitute single word transfer times so all connection times are deterministic with mean 1 or 10, both have been used in the experiments. Object code execution times for computation states have been chosen so that like operations require the same execution times. Again, the simulator is a set of P list merge processors written in SIMSCRIPT II.5.

To obtain reasonable program execution times for comparisons, the two source lists are taken to be 256 elements long (this choice also seems to indicate that a system

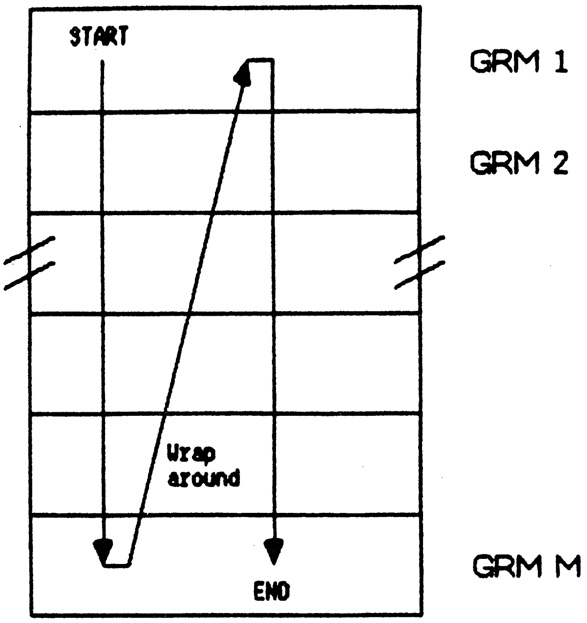


Figure 6.11 List layout in GRM's.

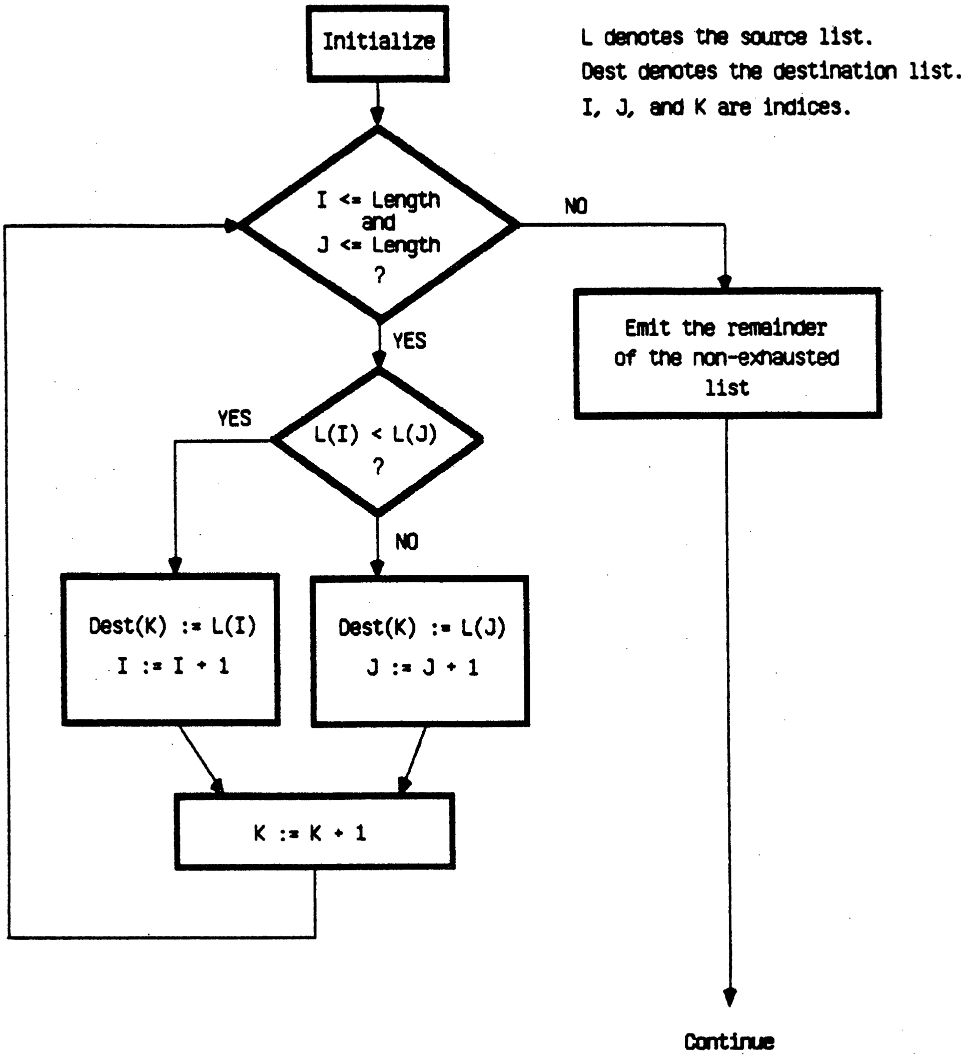


Figure 6.12 List merge flowchart.

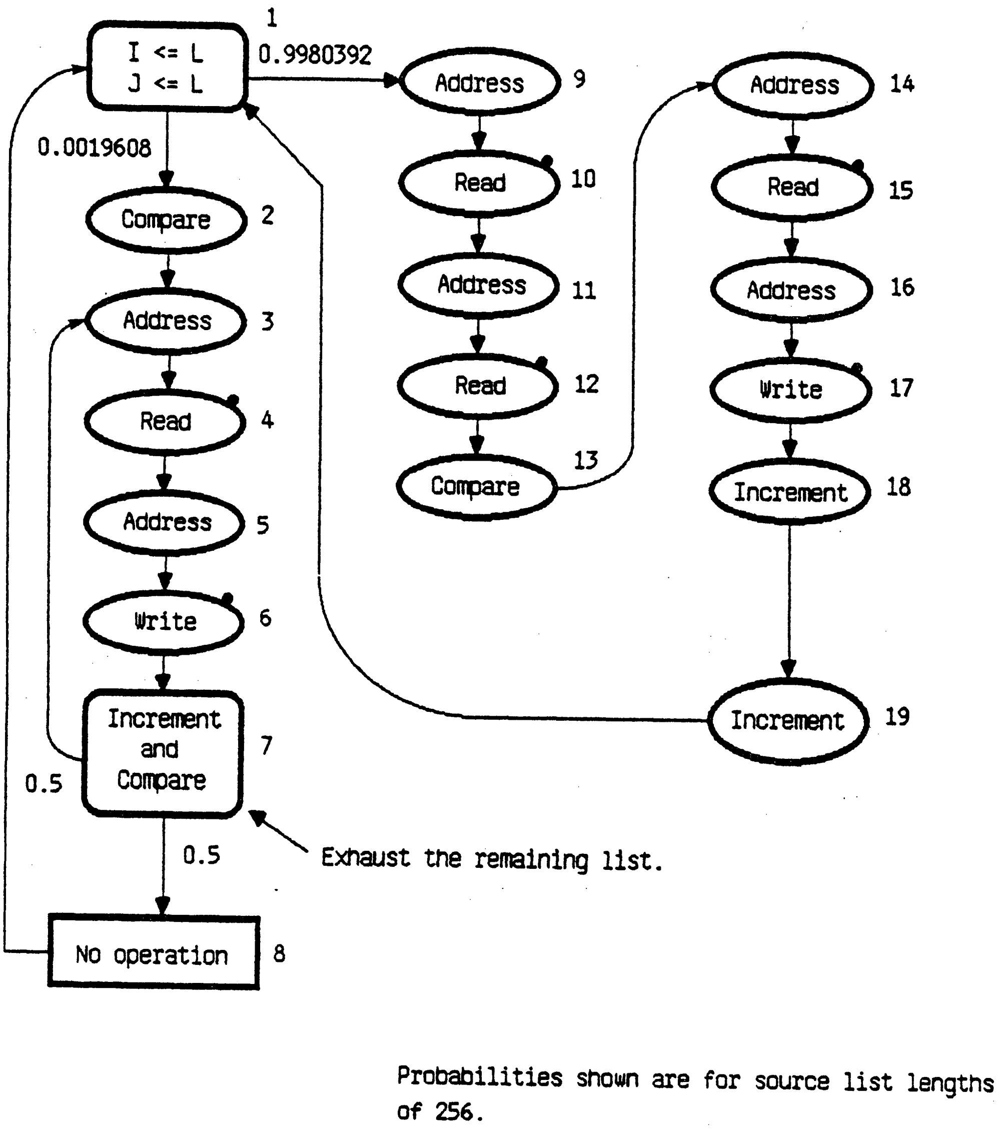


Figure 6.13 List merge PE state diagram.

"steady-state" may be reached relatively quickly, a few hundred GRM references seem to suffice in many circumstances). The simulator uses two lists of unformly distributed random numbers that have been presorted. Under these conditions [Knu73] describes an analysis of a merge algorithm, it is shown that the mean number of comparisons done on the two lists before one is exhausted is 2×256 - 2 = 510. This is then the mean number of loop executions evident for the list merge loop. 1/510 is the appropriate Bernoulli loop exiting probability used in the analysis. Since there are 512 elements in the result list, an average of 2 must be directly obtained from the source list that was not exhausted so the second loop (states 3-7) is executed on the average twice, its Bernoulli exit probability is 1/2.

Since this is a relatively simple example, connection and sojourn times are simple to obtain. Only one modeling point needs yet to be addressed: reference patterns. The list elements are interleaved so in the steady-state (e.e., where large lists are processed) all GRM's will be referenced equally often, hence according to the fractional argument we set $\eta_{p,*,m} = 1/M$ ($*$ denotes any appropriate index). In an effort to examine this reference pattern approximation, simulations have been done for actual patterns generated by the list merge program and synthetic references generated according to the model's fractional approximation (uniformly distributed randomly chosen GRM's).

To verify the model on various sized systems, several configurations have been examined. The results are shown on the next few pages for the short ($\overline{Y}_{p*m} = 1$) and long ($\overline{Y}_{p*m} = 10$) connection times with the actual and random reference patterns.

Short Connection Times, Actual Reference Patterns

P x M	$\overline{W}_{pm}$	ρ_m	Execution Time	
2 x 2	0.0	0.1818	11248	Simulated
	0.0278	0.1808	11281	Case 1 SMP
	0.0423	0.1804	11310	Case 2 SMP
	0.0096	0.1814	11244	G/G/1
	1.0	-0.2	-0.04	G/G/1% error
4 x 1	0.2506	0.6949	11760	Simulated
	0.2643	0.6937	11763	Case 1 SMP
	0.3040	0.6889	11844	Case 2 SMP
	0.2305	0.6978	11694	G/G/1
	-2.0	0.4	-0.6	G/G/1% error
4 x 2	0.1009	0.3570	11453	Simulated
	0.1265	0.3553	11482	Case 1 SMP
	0.1371	0.3547	11504	Case 2 SMP
	0.1171	0.3559	11463	G/G/1
	1.6	-0.3	0.1	G/G/1% error
4 x 4	0.0525	0.1804	11354	Simulated
	0.0650	0.1796	11357	Case 1 SMP
	0.0677	0.1795	11362	Case 2 SMP
	0.0703	0.1795	11367	G/G/1
	1.8	-0.5	0.1	G/G/1% error
8 x 4	0.1652	0.3531	11587	Simulated
	0.1623	0.3531	11555	Case 1 SMP
	0.1647	0.3529	11560	Case 2 SMP
	0.2006	0.3507	11633	G/G/1
	3.5	-0.7	0.4	G/G/1% error
8 x 8	0.0750	0.1802	11404	Simulated
	0.0792	0.1792	11386	Case 1 SMP
	0.0798	0.1792	11387	Case 2 SMP
	0.0931	0.1787	11414	G/G/1
	1.8	-0.8	0.1	G/G/1% error

Short Connection Times, Random Reference Patterns				
P x M	$\overline{W}_{pm}$	ρ_m	Execution Times	
2 x 2	0.0 0.0096 1.0	0.1818 0.1814 -0.2	11248 11244 -0.4	Simulated G/G/1 % error
4 x 1	0.2506 0.2305 -2.0	0.6949 0.6978 0.4	11760 11694 -0.6	Simulated G/G/1 % error
4 x 2	0.0770 0.1171 4.0	0.3583 0.3559 -0.7	11405 11463 0.5	Simulated G/G/1 % error
4 x 4	0.0290 0.0703 4.1	0.1808 0.1795 -0.7	11306 11367 0.5	Simulated G/G/1 % error
8 x 4	0.1641 0.2006 3.7	0.3528 0.3507 -0.6	11584 11633 0.4	Simulated G/G/1 % error
8 x 8	0.0696 0.0931 2.4	0.1795 0.1787 -0.4	11392 11414 0.2	Simulated G/G/1 % error

Since the same analysis results apply here as on the previous page, SMP waiting time results are not shown, they are as before.

Long Connection Times, Actual Reference Patterns

P x M	$\overline{W}_{pm}$	ρ_m	Execution Time	
2 x 2	2.7350	0.6164	33197	Simulated
	1.0595	0.6868	29701	Case 1 SMP
	1.7133	0.6573	31035	Case 2 SMP
	1.4860	0.6673	30571	G/G/1
	-9.8	8.3	-7.9	G/G/1 % error
4 x 1	26.4694	0.9900	81735	Simulated
	22.3081	1.0000	73048	Case 1 SMP
	22.6679	1.0000	73782	Case 2 SMP
	20.1935	1.0000	68734	G/G/1
	-17.2	1.0	-15.9	G/G/1 % error
4 x 2	10.1350	0.8423	48365	Simulated
	6.3781	1.0000	40551	Case 1 SMP
	6.6620	0.9920	41130	Case 2 SMP
	NC	NC	NC	G/G/1
	-17.2	17.8	-15.0	Case 2 SMP % error
4 x 4	4.0892	0.5654	35979	Simulated
	2.6344	0.6198	32914	Case 1 SMP
	2.7437	0.6150	33137	Case 2 SMP
	3.8492	0.5764	35392	G/G/1
	-1.7	1.9	-1.6	G/G/1 % error
8 x 4	12.1498	0.7690	52477	Simulated
	11.8418	0.7892	51697	Case 1 SMP
	11.9416	0.7861	51900	Case 2 SMP
	NC	NC	NC	G/G/1
	-0.0	2.2	-1.1	Case 2 SMP % error
8 x 8	5.0920	0.5303	38061	Simulated
	3.1802	0.5995	34027	Case 1 SMP
	3.2035	0.5987	34075	Case 2 SMP
	4.7424	0.5482	37214	G/G/1
	-2.3	3.4	-2.2	G/G/1 % error

NC denotes no convergence, in this event error given is computed using the case 2 SMP analysis.

Long Connection Times, Random Reference Patterns

P x M	$\overline{W}_{pm}$	ρ_m	Execution Time	
2 x 2	2.4202	0.6282	32555	Simulated
	1.4860	0.6673	30571	G/G/1
	-7.5	6.2	-6.1	% error
4 x 1	26.4694	0.9900	81740	Simulated
	20.1935	1.0000	68734	G/G/1
	-17.2	1.0	-15.9	% error
4 x 2	9.9947	0.8478	48059	Simulated
	6.6620	0.9920	41130	Case 2 SMP
	-16.7	17.0	-14.4	% error
4 x 4	3.8029	0.5746	35392	Simulated
	3.8492	0.5764	35392	G/G/1
	0.3	0.3	0	% error
8 x 4	11.7522	0.7838	51682	Simulated
	11.9416	0.7861	51900	Case 2 SMP
	0.9	0.3	0.4	% error
8 x 8	4.5972	0.5475	37038	Simulated
	4.7424	0.5482	37214	G/G/1
	1.0	0.1	0.5	% error

There are several points to notice about the list merge results. First note that in the short connection time case, a 2×2 system self-synchronizes. This has been noted in other 2×2 experiments not shown here. It has never been seen in systems other than 2×2's, although it is possible. Self synchronization does not occur in the long connection time case because equivalent state 1 mean sojourn time is not sufficient to allow connections by other PE's to be completed before the next request emission.

Secondly note that both the SMP and G/G/1 waiting time calculations are sufficiently accurate that either may be used. Reference time error is computed *including* the connection time because execution time error will be due to errors accrued in total GRM use time more directly than just waiting times. It is inappropriate to base error measure on a possibly small quantity (notice the magnitude of the waiting times for the short connection time case where the connection time is 1, they are very small). Hence error in GRM queueing times will include the connection time as:

$$\%\, Error = \frac{(\overline{W}_{pm} + \overline{Y}'_{pm})_{calc} - (\overline{W}_{pm} + \overline{Y}'_{pm})_{sim}}{(\overline{W}_{pm} + \overline{Y}'_{pm})_{sim}} \times 100\%$$

In the short connection time case, the range of execution time error is ±1% (computing error of the analytic prediction verses the simulated value). In the long connection time case, execution time error ranges from -16% to +1%. Note that in the long connection time case the G/G/1 waiting time based solution failed on compressor systems ($P > M$). It produced waiting time predictions that oscillated between extremes as the fixed point iteration proceeded. In this case the SMP based results were used.

Finally note that in both the actual and synthetic reference patterns, nearly the same simulated results were obtained. This fact supports the applicability of using reference fractions as time invariant probabilities.

The poorest performance of the waiting time calculations is present in compressor systems. This is expected because these systems display the most coupling between the processes $\{\tilde{Z}_p(t), t \geq 0\}$. This may be qualitatively seen by considering extreme cases: if there is one GRM, all PE's contend over it, thereby creating coupling between SMP's at time t; only one SMP may be in a GRM use state (connection phase) at time t. This exclusion influences the time limiting system state occupation probabilities. System state probabilities are not appropriately represented by simple products as in (3.3). The assumption that Poisson time statistics apply at emission times is inaccurate due to coupling.

At the other extreme, when there are a large (infinite) number of uniformly used GRM's, there will be negligible coupling between SMP's because they rarely interact in a temporal manner through GRM queueing; each may achieve its potential.

6.2. Synthetic Experiments

Three sets of results will be shown. The first set of data concerns a correlation test experiment where two simple loops with different characteristics are executed in an alternating sequence to determine the applicability of $\overline{W}_{psm} \simeq \overline{W}_{pm}$ and the Bernoulli loop approximation.

The second set of data examines an asymmetric example where each PE behaves as an ARP with highly asymmetric behavioral characteristics. The ARP description was chosen for its simplicity and the idea that fixing some stochastic properties while others are being studied aids interpretation of the results.

The third set of data shown here was generated by [Bha73]. It will be used to examine the applicability of the SMP model's physical analysis in a highly discrete environment. Differences in queueing disciplines should be evident here in that many of the previous models of memory interference -- of which [Hoo77] is a good example -- are based on a "random request selection" priority whereas the SMP physical analysis shown in chapter 4 is based on the FCFS discipline. The SMP model physical analysis does not include terms representing the occurrence of simultaneous events.

6.2.1. Simple Correlation Tests

Consider the state diagram shown in figure 6.14. Two loops consisting of computation and reference states are executed in sequence (repeatedly as state 5 provides a timing base for cycle time examination). The number of times each loop is executed is constant and is given by L_1 and L_2 respectively. That is, the simulator executes the states 1 and 2 exactly L_1 times and then proceeds to execute states 3 and 4 L_2 times. After completing the two loops it starts over again. All computation state sojourn and connection times are deterministic, this is often the most difficult case of CDF choices to handle accurately using a "continuous time" description that the G/G/1 equations exemplify.

To check those effects believed to be important, the connection times for state 2 and 4 are different and the loop counts are also different.

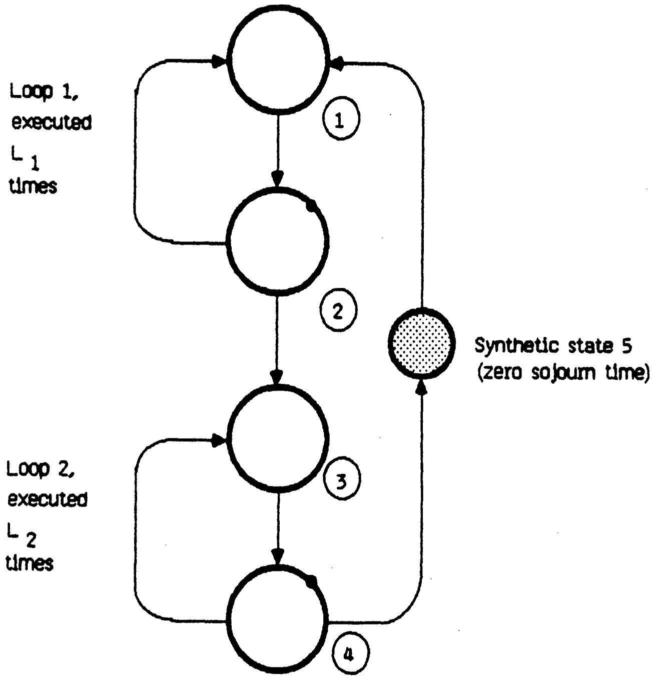


Figure 6.14 Simple correlation test PE state diagram.

Following are correlation test results, all simulated results are the average over three runs. Reference patterns are uniform over all GRM's, remaining parameters are also symmetric across all PE's and GRM's.

Example 1:

$$\overline{S}_{p1} = 1.0, \ \overline{S}_{p3} = 2.0$$

$$\overline{Y}_{p2m} = 5.0, \ \overline{Y}_{p4m} = 1.0$$

$$L_1 = 500, \ L_2 = 100$$

Example 2:

$$\overline{S}_{p1} = 1.0, \ \overline{S}_{p3} = 1.0$$

$$\overline{Y}_{p2m} = 100.0, \ \overline{Y}_{p4m} = 1.0$$

$$L_1 = 1, \ L_2 = 1$$

$P \times M$	$\overline{W}_{pm}$	ρ_m	$\overline{C}_{p,5,5}$		
2 x 2	1.115	0.6554	3968	Simulated	Example 1
	.573	0.7135	3644	Case 1 SMP	
	.887	0.6785	3832	Case 1 SMP	
	.938	0.6731	3863	G/G/1	
4 x 4	1.8769	0.5886	4408	Simulated	
	1.3193	0.6203	4192	Case 1 SMP	
	1.3690	0.6159	4221	Case 2 SMP	
	1.9210	0.5711	4553	G/G/1	
8 x 8	2.2653	0.5589	4662	Simulated	
	1.6292	0.6078	4278	Case 1 SMP	
	1.6397	0.6069	4284	Case 2 SMP	
	2.3685	0.5507	4721	G/G/1	
16 x 16	2.4498	0.5456	4775	Simulated	
	1.7560	0.5972	4354	Case 1 SMP	
	1.7584	0.5970	4355	Case 2 SMP	
	2.5359	0.5392	4822	G/G/1	
2 x 2	21.7	0.6905	146	Simulated	Example 2
	21.9	0.6879	147	Case 1 SMP	
	23.7	0.6714	150	Case 2 SMP	
	32.6	0.6004	168	G/G/1	
4 x 4	34.4	0.5870	170	Simulated	
	32.8	0.5985	169	Case 1 SMP	
	33.0	0.5969	169	Case 2 SMP	
	42.3	0.5383	188	G/G/1	
8 x 8	40.4	0.5474	185	Simulated	
	38.1	0.5635	179	Case 1 SMP	
	38.1	0.5632	179	Case 2 SMP	
	48.2	0.5062	200	G/G/1	
16 x 16	44.5	0.5223	197	Simulated	
	41.6	0.5419	186	Case 1 SMP	
	41.6	0.5418	186	Case 2 SMP	
	60.2	0.4519	224	G/G/1	

Results seem to indicate that the Bernoulli loop approximation is sufficient for first moment predictions. The waiting time average from the simulator is $\overline{W}_{pm}$ accumulated during the simulations, it is not $\overline{W}_{psm}$. The cycle time simulated should reflect inaccuracies induced by the state dependence, if it is not detectable from state cycle times, then it is sufficient to use $\overline{W}_{pm}$. The second example does display cycle time error that may be attributable to inaccuracies in the state independence approximation the error is not excessive. Contrary examples might be devised but in typical circumstances the state invariant waiting time calculations seem sufficient.

6.2.2. Asymmetric Tests

In this example PE's behave as ARP's with vastly different behavior characteristics. The example was chosen to display errors in the waiting time calculations in a system whose behavior is difficult to predict accurately.

Since systems with $P > M$ are most difficult to handle accurately, the example considers a 7×4 system. To check asymmetric calculations, some PE's "prefer" a certain GRM, in particular one PE uses exclusively one GRM. This preference was chosen to check the dependence of a PE's waiting time on its own use of the GRM, i.e., the dependence of $\overline{W}_{pm}$ on ρ_{pm}. In this case, PE 5 uses only GRM 1.

CDF's have also been varied between experiments, the following CDF's have been used: deterministics; exponentials and Erlangs; and uniform CDF's.

Threshold schemes have also been changed and the calculations done for two thresholding (requestor counting) schemes: the one described by equations (4.12) and (4.13); and a simple scheme $|Q_{pm}| = |\#_{pm}|$. Parameters and results are shown on the next few pages.

For all three test cases, the GRM reference matrix is:

$$\eta' = \begin{array}{cc} & \xrightarrow{\text{m}} \\ \text{p}\downarrow & \begin{bmatrix} 0.0 & 0.5 & 0.25 & 0.25 \\ 0.0 & 0.0 & 0.5 & 0.5 \\ 0.1 & 0.2 & 0.3 & 0.4 \\ 0.4 & 0.3 & 0.2 & 0.1 \\ 1.0 & 0.0 & 0.0 & 0.0 \\ 0.2 & 0.5 & 0.1 & 0.2 \\ 0.7 & 0.2 & 0.05 & 0.05 \end{bmatrix} \end{array}$$

The *first* case equivalent connection time CDF's are given by:

$$Y'(t) = \begin{array}{cc} & \xrightarrow{\text{m}} \\ \text{p}\downarrow & \begin{bmatrix} & E(98,6) & E(69,8) & E(46,3) \\ & & E(26,3) & E(90,10) \\ E(96,7) & E(76,5) & E(23,6) & E(82,5) \\ E(94,9) & E(20,5) & E(58,3) & E(76,4) \\ E(97,8) & & & \\ E(57,4) & E(45,10) & E(65,3) & E(11,1) \\ E(42,10) & E(18,6) & E(24,2) & E(91,5) \end{bmatrix} \end{array}$$

$E(\alpha,\kappa)$ denotes an Erlang CDF, with mean α and κ stages of exponential service.

In *cases 2 and 3,* the following connection time CDF's apply:

$$\mathrm{Y'(t)} = \begin{array}{cc} & \xrightarrow{\mathrm{m}} \\ \mathrm{p}\downarrow & \left[\begin{array}{cccc} & \mathrm{u(t\text{-}98)} & \mathrm{u(t\text{-}69)} & \mathrm{u(t\text{-}46)} \\ & & \mathrm{u(t\text{-}26)} & \mathrm{u(t\text{-}90)} \\ \mathrm{u(t\text{-}96)} & \mathrm{u(t\text{-}76)} & \mathrm{u(t\text{-}23)} & \mathrm{u(t\text{-}82)} \\ \mathrm{u(t\text{-}94)} & \mathrm{u(t\text{-}20)} & \mathrm{u(t\text{-}58)} & \mathrm{u(t\text{-}76)} \\ \mathrm{u(t\text{-}97)} & & & \\ \mathrm{u(t\text{-}57)} & \mathrm{u(t\text{-}45)} & \mathrm{u(t\text{-}65)} & \mathrm{u(t\text{-}11)} \\ \mathrm{u(t\text{-}42)} & \mathrm{u(t\text{-}18)} & \mathrm{u(t\text{-}24)} & \mathrm{u(t\text{-}91)} \end{array}\right] \end{array}$$

$u(t)$ is the unit step function: $u(t<0) = 0$, $u(t \geq 0) = 1$.

The state 1 (equivalent) computation time CDF's are given by:

$$\mathrm{S}(t) = \begin{array}{cc} & \begin{array}{ccc} \text{Case 1, State 1} & \text{Case 2, State 1} & \text{Case 3, State 1} \end{array} \\ \mathrm{p}\downarrow & \left[\begin{array}{ccc} \mathrm{U(29,83)} & \mathrm{E(56,1)} & \mathrm{u(t\text{-}56)} \\ \mathrm{E(23,4)} & \mathrm{E(23,1)} & \mathrm{u(t\text{-}23)} \\ \mathrm{E(57,9)} & \mathrm{E(57,1)} & \mathrm{u(t\text{-}57)} \\ \mathrm{U(25,37)} & \mathrm{E(31,1)} & \mathrm{u(t\text{-}31)} \\ \mathrm{U(30,95)} & \mathrm{E(62.5,1)} & \mathrm{u(t\text{-}62.5)} \\ \mathrm{U(55,68)} & \mathrm{E(61.5,1)} & \mathrm{u(t\text{-}61.5)} \\ \mathrm{E(19,2)} & \mathrm{E(19,1)} & \mathrm{u(t\text{-}19)} \end{array}\right] \end{array}$$

$\mathrm{U}(\alpha,\beta)$ denotes a uniform distribution between α and β.

Case 1 is composed of Erlang and uniform CDF's; case 2 is composed of exponential computation times and constant connection times; case 3 displays *only* deterministic computation and connection times. The results are shown next.

Case 1: Simulated and G/G/1 results

Thresholding is given by (4.12) and (4.13).

	GRM 1	GRM 2	GRM 3	GRM 4		
PE 1		14.4 19.4 4.4	10.9 10.8 0.1	65.8 92.4 23.8	Simulated G/G/1 Soln. % error	$\overline{W}_{pm}$
PE 2			11.5 15.5 10.7	34.7 39.7 4.0	Simulated G/G/1 Soln. % error	
PE 3	128.8 144.9 7.2	31.7 37.2 5.1	14.8 14.6 -0.5	50.9 52.8 1.4	Simulated G/G/1 Soln. % error	
PE 4	105.6 97.4 -4.1	37.3 35.9 -2.4	13.4 11.6 -2.5	67.7 95.8 19.6	Simulated G/G/1 Soln. % error	
PE 5	78.8 75.8 -1.7				Simulated G/G/1 Soln. % error	
PE 6	131.8 150.2 9.7	32.3 34.4 2.7	14.5 13.2 -1.6	74.4 108.1 39.5	Simulated G/G/1 Soln. % error	
PE 7	115.1 113.2 -1.2	38.9 36.2 -4.7	17.5 14.8 -6.5	74.8 102.2 16.5	Simulated G/G/1 Soln. % error	
	.9699 .9724 -0.3	.5973 .5712 -4.4	.3990 .3824 -4.2	.7930 .7682 -3.1	Simulated G/G/1 Soln. % error	ρ_m
	73.3 67.2 -8.3	56.3 49.5 -12.1	38.2 38.2 0.0	72.4 71.9 -0.7	Simulated G/G/1 Soln. % error	$\overline{X}_m$
	.0132 .0132 0	.0106 .0103 -2.8	.0104 .0100 -3.8	.0110 .0106 -3.6	Simulated G/G/1 Soln. % error	λ_m^i
	.697 .672 -3.6	.824 .805 -2.3	1.06 .799 -24.6	.647 .793 22.6	Simulated G/G/1 Soln. % error	$V_{M_m}^2$

Case 1: SMP case 1 and case 2 results

Thresholds are given by (4.12) and (4.13).

	GRM 1	GRM 2	GRM 3	GRM 4		
PE 1		13.9	10.0	41.4	SMP case 1	$\overline{W}_{pm}$
		13.8	10.0	41.8	SMP case 2	
PE 2			11.5	23.6	SMP case 1	
			11.2	22.8	SMP case 2	
PE 3	75.0	26.6	13.4	31.5	SMP case 1	
	78.3	26.6	13.7	32.7	SMP case 2	
PE 4	52.2	29.0	10.9	40.7	SMP case 1	
	54.3	29.0	10.8	40.9	SMP case 2	
PE 5	44.6				SMP case 1	
	45.0				SMP case 2	
PE 6	78.0	25.1	12.5	41.5	SMP case 1	
	81.8	26.2	12.5	42.0	SMP case 2	
PE 7	48.7	28.9	14.0	40.6	SMP case 1	
	58.5	29.1	14.0	40.8	SMP case 2	
	1.0 (1.20)	.66073	.4373	.8682	SMP case 1	ρ_m
	1.0 (1.18)	.6558	.4359	.8662	SMP case 2	
	67.2	49.5	38.2	71.9	SMP case 1	$\overline{X}_m$
	67.2	49.5	38.2	71.9	SMP case 2	
	.0170	.0124	.0114	.0120	SMP case 1	λ'_m
	.0165	.0122	.0114	.0120	SMP case 2	
	.881	.980	.846	.866	SMP case 1	$V^2_{M_m}$
	.819	.968	.843	.863	SMP case 2	
	-3.6	-2.3	-24.6	22.6	% error	

Error has not been computed due to the clear inaccuracies when compared to the G/G/1 based waiting time predictions. Parenthesized utilizations are those that result from using utilization equations with the solution waiting times, clearly the waiting times are too small in these cases so the actual utilization predictions should be 1.0.

Case 2: Simulated and G/G/1 results

Thresholds are given by (4.12) and (4.13).

	GRM 1	GRM 2	GRM 3	GRM 4		
PE 1		15.3 17.3 1.8	9.3 7.8 -1.9	62.0 87.4 23.5	Simulated G/G/1 Soln. % error	$\overline{W}_{pm}$
PE 2			10.2 12.9 7.5	32.4 33.5 0.9	Simulated G/G/1 Soln. % error	
PE 3	126.3 144.7 8.3	30.6 33.2 2.4	12.4 11.3 -3.1	49.3 48.0 -1.0	Simulated G/G/1 Soln. % error	
PE 4	102.0 90.8 -5.7	34.0 32.0 -3.7	11.7 9.3 -3.4	64.3 91.1 19.1	Simulated G/G/1 Soln. % error	
PE 5	82.4 69.3 -7.3				Simulated G/G/1 Soln. % error	
PE 6	126.2 149.9 12.9	28.8 29.6 1.1	12.7 10.5 -2.8	66.4 103.6 48.1	Simulated Calc % error	
PE 7	114.2 106.8 -4.7	35.2 32.1 -5.8	15.2 11.7 -8.9	68.5 97.7 18.3	Simulated G/G/1 Soln. % error	
	.9713 1.000 3.0	.6025 .5835 -3.2	.4062 .3937 -3.1	.8053 .7944 -1.4	Simulated G/G/1 Soln. % error	ρ_m
	73.2 67.2 -8.2	55.9 49.5 -11.4	38.2 38.2 0	72.2 71.9 -0.4	Simulated G/G/1 Soln. % error	$\overline{X}_m$
	.0133 .0137 3.0	.0108 .0105 -2.8	.0106 .0104 -1.9	.0112 .0109 -2.7	Simulated G/G/1 Soln. % error	λ_m^1
	.630 .669 6.2	.846 .822 -2.8	1.02 .789 -22.6	.628 .788 25.5	Simulated G/G/1 Soln. % error	$V_{M_m}^2$

Case 2: SMP case 1 and case 2 results

Thresholds are given by (4.12) and (4.13).

	GRM 1	GRM 2	GRM 3	GRM 4		
PE 1		11.7	7.4	36.1	SMP case 1	$\overline{W}_{pm}$
		11.6	7.5	36.5	SMP case 2	
PE 2			9.2	18.9	SMP case 1	
			8.9	18.2	SMP case 2	
PE 3	68.4	22.7	10.5	27.3	SMP case 1	
	71.5	22.7	10.8	28.4	SMP case 2	
PE 4	47.5	24.8	8.5	35.4	SMP case 1	
	49.3	24.7	8.7	35.6	SMP case 2	
PE 5	40.9				SMP case 1	
	41.2				SMP case 2	
PE 6	71.3	21.2	10.0	36.0	SMP case 1	
	74.8	22.1	10.0	36.4	Calc	
PE 7	44.4	24.7	11.1	35.3	SMP case 1	
	53.0	24.8	11.1	35.4	SMP case 2	
	1.0 (1.24)	.6784	.4507	.8970	SMP case 1	ρ_m
	1.0 (1.22)	.6737	.4494	.8949	SMP case 2	
	67.2	49.5	38.2	71.9	SMP case 1	$\overline{X}_m$
	67.2	49.5	38.2	71.9	SMP case 2	
	.0176	.0128	.0118	.0124	SMP case 1	$\lambda_m^{'}$
	.0171	.0126	.0118	.0124	SMP case 2	
	.899	.979	.830	.846	SMP case 1	$V_{M_m}^2$
	.840	.967	.826	.844	SMP case 2	

Case 3: Simulated and G/G/1 results

Thresholds are given by (4.12) and (4.13).

	GRM 1	GRM 2	GRM 3	GRM 4		
PE 1		12.2	8.4	61.1	Simulated	$\overline{W}_{pm}$
		17.0	7.6	86.9	G/G/1 Soln.	
		4.4	-1.0	24.1	% error	
PE 2			9.75	32.3	Simulated	
			12.7	32.9	G/G/1 Soln.	
			8.1	0.5	% error	
PE 3	126.2	28.1	12.3	48.8	Simulated	
	143.6	32.4	11.0	47.5	G/G/1 Soln.	
	7.8	4.1	-3.7	-1.0	% error	
PE 4	103.2	32.1	10.3	63.3	Simulated	
	90.1	31.0	9.0	90.7	G/G/1 Soln.	
	-6.6	-2.1	-1.9	19.7	% error	
PE 5	87.3				Simulated	
	70.6				G/G/1 Soln.	
	-9.1				% error	
PE 6	130.0	30.5	12.8	70.5	Simulated	
	148.7	29.0	10.3	103.3	Calc	
	-10.0	-2.0	-3.2	40.2	% error	
PE 7	104.9	33.9	16.1	68.9	Simulated	
	97.9	31.1	11.5	97.0	G/G/1 Soln.	
	-4.8	-5.4	-11.5	17.6	% error	
	.9723	.6115	.4087	.8072	Simulated	ρ_m
	1.00	.5862	.3954	.7988	G/G/1 Soln.	
	2.8	-4.1	-3.3	-1.0	% error	
	74.5	56.1	38.1	72.4	Simulated	$\overline{X}_m$
	67.2	49.5	38.2	71.9	G/G/1 Soln.	
	-9.8	-11.8	0.3	-0.7	% error	
	.0134	.0109	.0104	.0112	Simulated	$\lambda_m^{'}$
	.0139	.0106	.0104	.0110	G/G/1 Soln.	
	3.7	-2.8	0	-1.8	% error	
	.619	.777	.991	.604	Simulated	$V_{M_m}^2$
	.647	.795	.772	.768	G/G/1 Soln.	
	4.5	2.3	-22.1	27.2	% error	

Case 3: SMP case 1 and case 2 results

Thresholds are given by (4.12) and (4.13).

	GRM 1	GRM 2	GRM 3	GRM 4		
PE 1		11.7	7.4	36.0	SMP case 1	$\overline{W}_{pm}$
		11.6	7.4	36.4	SMP case 2	
PE 2			9.1	19.0	SMP case 1	
			8.9	18.1	SMP case 2	
PE 3	67.9	22.6	10.5	27.2	SMP case 1	
	71.3	22.7	10.7	28.4	SMP case 2	
PE 4	47.0	24.6	8.5	35.4	SMP case 1	
	49.1	24.7	8.6	35.6	SMP case 2	
PE 5	40.5				SMP case 1	
	41.0				SMP case 2	
PE 6	70.7	21.0	9.9	35.9	SMP case 1	
	74.6	22.0	10.0	36.4	SMP case 2	
PE 7	43.3	24.5	11.0	35.2	SMP case 1	
	52.6	24.7	11.1	35.4	SMP case 2	
	1.0 (1.24)	.6793	.4512	.8980	SMP case 1	ρ_m
	1.0 (1.22)	.6741	.4497	.8955	SMP case 2	
	67.2	49.5	38.2	71.9	SMP case 1	$\overline{X}_m$
	67.2	49.5	38.2	71.9	SMP case 2	
	.0177	.0128	.0118	.0124	SMP case 1	λ^i_m
	.0172	.0126	.0118	.0124	SMP case 2	
	.880	.955	.813	.829	SMP case 1	$V^2_{M_m}$
	.799	.939	.807	.825	SMP case 2	

Case 1: Simulated and G/G/1 results with $Q_{pm} = \#_{pm}$

	GRM 1	GRM 2	GRM 3	GRM 4		
PE 1		14.4 19.8 4.8	10.9 10.8 -.1	65.8 88.3 20.1	Simulated G/G/1 Soln. % error	$\overline{W}_{pm}$
PE 2			11.5 15.5 10.7	34.7 44.3 7.7	Simulated G/G/1 Soln. % error	
PE 3	128.8 216.7 39.1	31.7 37.4 5.3	14.8 14.4 -1.1	50.9 51.7 0.6	Simulated G/G/1 Soln. % error	
PE 4	105.6 100.8 -2.4	37.3 42.7 9.4	13.4 11.6 -2.5	67.7 91.1 16.3	Simulated G/G/1 Soln. % error	
PE 5	78.8 76.7 -1.2				Simulated G/G/1 Soln. % error	
PE 6	131.8 224.1 48.9	32.3 34.0 2.2	14.5 13.2 -1.6	74.4 102.1 32.4	Simulated G/G/1 Soln. % error	
PE 7	115.1 107.9 -4.6	38.9 42.9 6.6	17.5 14.7 -6.7	74.8 96.2 12.9	Simulated G/G/1 Soln. % error	
	.9699 .9631 -0.7	.5973 .5597 -6.3	.3990 .3750 -6.0	.7930 .7514 -5.2	Simulated G/G/1 Soln. % error	ρ_m
	73.3 67.2 -8.3	56.3 49.5 -12.1	38.2 38.2 0.0	72.4 71.9 -0.7	Simulated G/G/1 Soln. % error	$\overline{X}_m$
	.0132 .0131 -0.8	.0106 .0100 -5.7	.0104 .0098 -5.8	.0110 .0103 -6.4	Simulated G/G/1 Soln. % error	λ'_m
	.697 .704 1.0	.824 .838 1.7	1.06 .818 -22.8	.647 .817 26.3	Simulated G/G/1 Soln. % error	$V^2_{M_m}$

Case 2: Simulated and G/G/1 results with $Q_{pm} = \#_{pm}$

	GRM 1	GRM 2	GRM 3	GRM 4		
PE 1		15.3	9.3	62.0	Simulated	$\overline{W}_{pm}$
		17.6	7.8	83.2	G/G/1 Soln.	
		2.0	-1.9	19.6	% error	
PE 2			10.2	32.4	Simulated	
			13.0	37.7	G/G/1 Soln.	
			7.7	4.3	% error	
PE 3	126.3	30.6	12.4	49.3	Simulated	
	216.4	33.4	11.1	46.9	G/G/1 Soln.	
	40.5	2.6	-3.7	-1.8	% error	
PE 4	102.0	34.0	11.7	64.3	Simulated	
	92.0	38.2	9.3	86.2	G/G/1 Soln.	
	-5.4	7.8	-3.4	15.6	% error	
PE 5	82.4				Simulated	
	70.5				G/G/1 Soln.	
	-6.6				% error	
PE 6	126.2	28.8	12.7	66.4	Simulated	
	223.7	29.3	10.5	97.3	G/G/1 Soln	
	53.2	0.7	-2.8	39.9	% error	
PE 7	114.2	35.2	15.2	68.5	Simulated	
	101.6	38.3	11.7	91.5	G/G/1 Soln.	
	-8.1	5.8	-8.9	14.4	% error	
	.9713	.6025	.4062	.8053	Simulated	ρ_m
	.9934	.5718	.3866	.7774	G/G/1 Soln.	
	2.3	-5.1	-4.8	-3.5	% error	
	73.2	55.9	38.2	72.2	Simulated	$\overline{X}_m$
	67.2	49.5	38.2	71.9	G/G/1 Soln.	
	-8.2	-11.4	0	-0.4	% error	
	.0133	.0108	.0106	.0112	Simulated	λ'_m
	.0136	.0103	.0102	.0106	G/G/1 Soln.	
	2.3	-4.6	-3.8	-5.4	% error	
	.630	.846	1.02	.628	Simulated	$V^2_{M_m}$
	.696	.854	.806	.810	G/G/1 Soln.	
	10.5	0.9	-21.0	29.0	% error	

Case 3: Simulated and G/G/1 results with $Q_{pm} = \#_{pm}$

	GRM 1	GRM 2	GRM 3	GRM 4		
PE 1		12.2 17.4 4.7	8.38 7.6 -1.0	61.1 82.6 20.1	Simulated G/G/1 Soln. % error	$\overline{W}_{pm}$
PE 2			9.75 12.8 8.5	32.3 37.1 3.9	Simulated G/G/1 Soln. % error	
PE 3	126.2 214.8 39.9	28.1 32.7 4.4	12.3 11.0 -3.7	48.8 46.4 -1.8	Simulated G/G/1 Soln. % error	
PE 4	103.2 91.3 -6.0	32.1 37.1 9.6	10.3 9.1 -1.8	63.3 85.7 16.1	Simulated G/G/1 Soln. % error	
PE 5	87.3 71.9 -8.4				Simulated G/G/1 Soln. % error	
PE 6	130.0 222.0 -49.2	30.5 28.7 -2.4	12.8 10.3 -3.2	70.5 96.8 32.3	Simulated G/G/1 Soln. % error	
PE 7	104.9 93.4 -7.8	33.9 37.2 6.4	16.1 11.4 11.7	68.9 90.7 13.6	Simulated G/G/1 Soln. % error	
	.9723 1.001 2.8	.6115 .5745 -6.1	.4087 .3883 -5.0	.8072 .7815 -3.2	Simulated G/G/1 Soln. % error	ρ_m
	74.5 67.2 -9.8	56.1 49.5 -11.8	38.1 38.2 0.3	72.4 71.9 -0.7	Simulated G/G/1 Soln. % error	$\overline{X}_m$
	.0134 .0138 3.0	.0109 .0104 -4.6	.0104 .0102 -1.9	.0112 .0107 -4.5	Simulated G/G/1 Soln. % error	λ_m^i
	.619 .673 8.7	.777 .826 5.9	.991 .788 -20.5	.604 .790 30.8	Simulated G/G/1 Soln. % error	$V_{M_m}^2$

When the thresholding scheme is $Q_{pm} = \#_{pm}$, the G/G/1 results are poor when compared to the results obtained when using the thresholding scheme in (4.12) and (4.13).

6.3. Comparison with a Previous Synchronous Model

Consider a $P \times M$ system where each PE's behavior is represented by an ARP with constant, unit valued, computation state sojourn and connection times. Requests are directed uniformly over all GRM's. The table shown below displays results from the SMP model and the results from [Hoo77], the simulated confidence interval is from [Bha73] as shown in [Hoo77].

$P \times M$	ρ_m	90 % Con. Int.	From [Hoo77]	SMP model
4 × 4	.4463	.4549, .4569	.4480	G/G/1
	.4588			SMP case 1
	.4569			SMP case 2
4 × 8	.2360	.2391, .2406	.2377	G/G/1
	.2392			SMP case 1
	.2390			SMP case 2
8 × 4	.7167	.7115, .7302	.7447	G/G/1
	.7864			SMP case 1
	.7840			SMP case 2
8 × 8	.4334	.4320, .4418	.4403	G/G/1
	.4508			SMP case 1
	.4504			SMP case 2

The results are comparable even thought the physical analysis done in chapter 4 does not include effects due to synchronous events such as request collisions. Note that the results are tolerant of the different queueing disciplines that were used, in [Hoo77] a random service priority was assumed -- the simulation data represents this situation. The physical analysis in chapter 4 assumed a FCFS service discipline but the difference in results is small. This is to be expected (approximately) because mean queueing times are independent of work conserving and non-job dependent service priorities in ideal, open queueing stations [GrH74]. The effects here are comparable although different service disciplines may not yield *exactly* the same mean waiting times because of dependence on higher moments (which are affected by the queueing discipline). For example, since different service disciplines affect second moments of queueing times, the coefficient of variation of queue interarrival times is affected by service disciplines, which in turn affects the mean queueing times (4.24). The service discipline might be neglected or assumed to be FCFS because of the mean value approximate equivalence.

6.4. Conclusions

Experimental data has been shown for many system configurations. The accuracy of the model calculations described in chapters 3 through 5 have been found to be accurate. Considering the complexity of the problem, the accuracy is quite acceptable. It is believed that enough experimental data has been shown to establish the model calculations as reasonably accurate.

By considering data from both the independent SMP and the G/G/1 calculations for waiting times it may be concluded that the G/G/1 calculations seem to give the best consistent set of predictions. They work well in both asymmetric and symmetric situations, they do fail to converge in compressor systems with high GRM utilizations. In these cases, the independent SMP waiting time calculations may be used with reasonable success.

CHAPTER 7

A CACHE MODELING EXAMPLE

This chapter describes a processor/cache memory model. A simple instruction execution atom is formed that may be used in further program modeling efforts; such as in describing subprograms, etc., when program segments may be parameterized in terms of instructions as atoms. The cache model is appropriate for systems where instructions and data are stored in global memory and local cache memory is associated with each processor. Note that a shared, circuit switched, bus with global memory attached to it behaves as a single GRM. Hence the analysis described here applies directly to simple single shared bus systems with distributed caches.

Parameters of the model include: the fraction of memory references that fetch instructions, the fraction that fetch data; the fraction of instructions that read operands, the fraction that write results, etc.; cache memory hit ratios; and firmware execution timing statistics.

The performance measure of primary importance is the mean instruction execution rate. Utilizations may be used to show that, for example, there is enough GRM utilization not used by processors for system devices not included in the processor set, such as I/O channels that use global memory. Utilizations are not primary in describing system operation speed. I/O channels may be included in the system model if the system evaluator designates appropriate device activity descriptions. If all active devices are included in the model, utilizations become secondary in importance. The model description will be pursued, solutions are not sought.

7.1. PE Cache Memory Organization

With each processor will be associated a cache memory which stores local data (that is allowed to be cached) and PE instructions, see figure 7.1. The finite capacity of PE cache memory may require cache content swapping. The memory management unit examines addresses generated by processor hardware and does any virtual memory translation and cache or global memory request control required. Virtual memory behavior will be ignored in this chapter.

Processors fetch instructions from local cache memory or from global memory when a cache miss occurs. Data references made by processors are either directed toward cache memory or are directed to global memory when: data from global data structures is not allowed to be cached, as in the case of global data structures; or when the contents of the referenced location are not presently contained in cache memory. A global cache memory (i.e., one cache that is used by all PE's), could be modeled using a subset of the GRM's. The reference pattern parameters (η_{psm}) may be used to model such configurations.

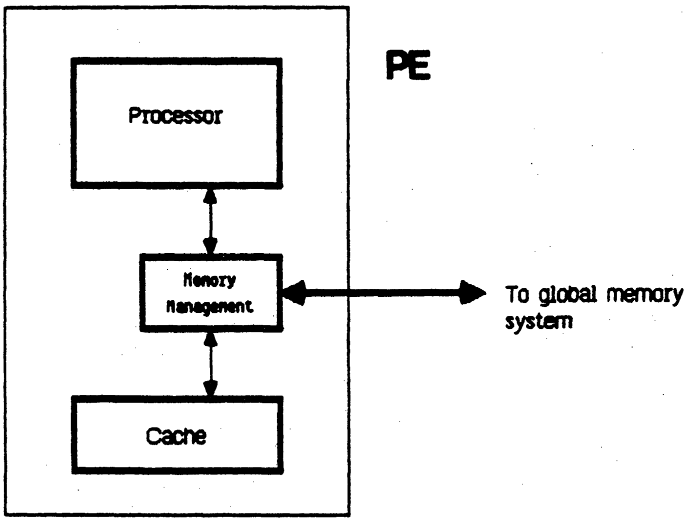


Figure 7.1 Processor/cache configuration.

Consider cache content replacement policies. The term "write-through" pertains to the cache maintenance policy where any memory write operation takes place in cache (if the referenced word is contained in cache) *and* global memory. Hence a write "through" the cache takes place. If the referenced word is not in cache, then a write operation will take place only in global memory, no line fetch or replacement will be done, see [Smi82]. In this case any line in cache may be found in an identical form in global memory. In a multiprocessor, this property may only be obtained if the cache coherence problem is solved in some manner. It will be assumed that data is tagged as cacheable or non-cacheable, and non-cacheable data may only be stored in global memory, so the coherency problem will not be important.

Another maintenance policy also considered will be termed "copy-back" [Smi82]. In this policy, cache lines needing replacement upon cache misses are copied back into global memory before new lines take their position in cache (there may certainly be some fetch overlapping in the physical implementation, hence bounds may be the only obtainable results, see the later discussion). Processors write cacheable data only into cache. Furthermore, if the referenced word is not in cache at the time, it will be read into cache before the write proceeds (the write into cache may actually take place concurrent with the cache line load, such details will be ignored in this example). This policy causes one or two global memory transfers to take place upon a cache miss: one if the line to be replaced has not been modified since being read into cache, its image exists in global memory so it need not be copied back; or two when the line to be replaced has been modified since being read into cache, its image in global memory is no longer valid.

When a line is fetched from global memory by the memory management unit L words are moved at a time, this group of words is called a cache line (appropriate read through to the processor might be done). This line fetch may be done in many ways in the context of the memory system considered.

In the two extremes considered here, a line may be moved in a single circuit switched connection to an appropriate GRM, or a line may be moved with one request sent to L GRM's (of which there may be kL, k an integer) for a single word transfer to/from each GRM simultaneously. These two forms of line transfers will be termed line fetching and scatter fetching respectively. An interesting use of the model is the comparison of these two techniques for implementing cache/global memory interactions.

Appropriate instruction execution atoms are shown in figure 7.2. Entries to state 1 designate the initiation of new instructions, hence the mean time between entries to state 1 ($\overline{C}_{p11}$) represents the mean instruction execution time for a given PE.

For the purposes of this example, consider the processor to be a one address machine so that associated with an instruction execution is at most one data reference. *More elaborate instruction execution models may be developed based on a particular processor design.* For example, multi-memory operand instructions could be modeled by associating with a sequence of k global memory references, k reference states in the PE

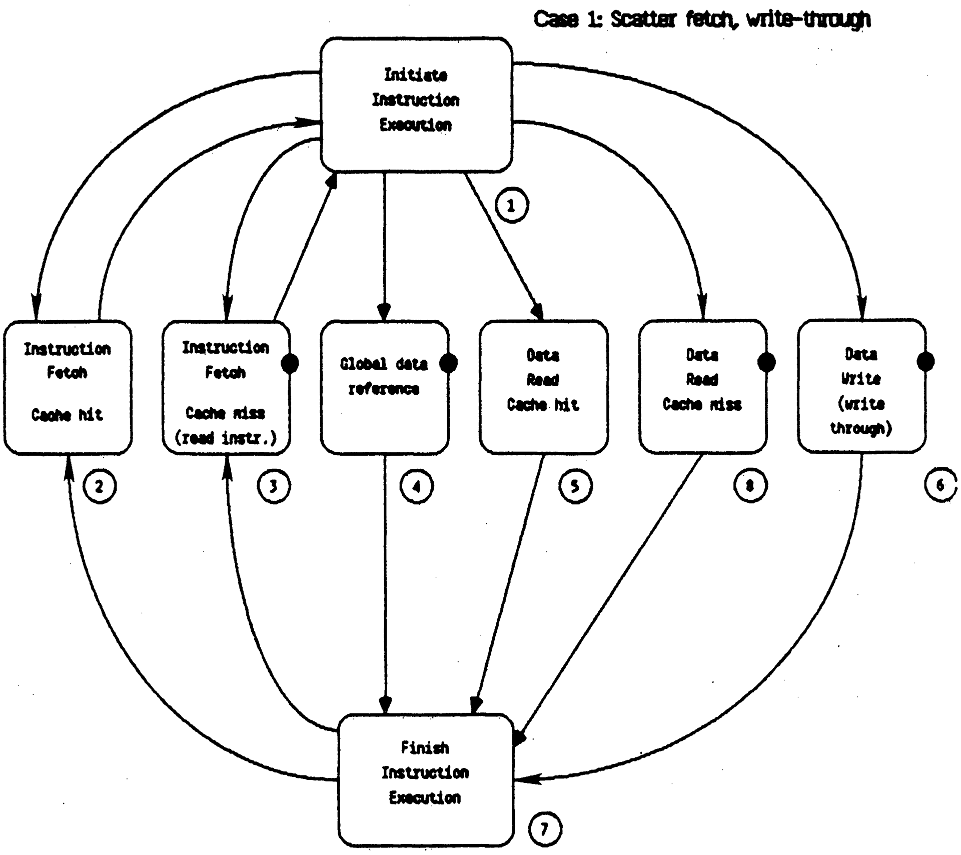

Case 1: Scatter fetch, write-through
Initiate Instruction Execution
1
Instruction Fetch Cache hit
2
Instruction Fetch Cache miss (read instr.)
3
Global data reference
4
Data Read Cache hit
5
Data Read Cache miss
8
Data Write (write through)
6
Finish Instruction Execution
7

Figure 7.2 PE descriptions.

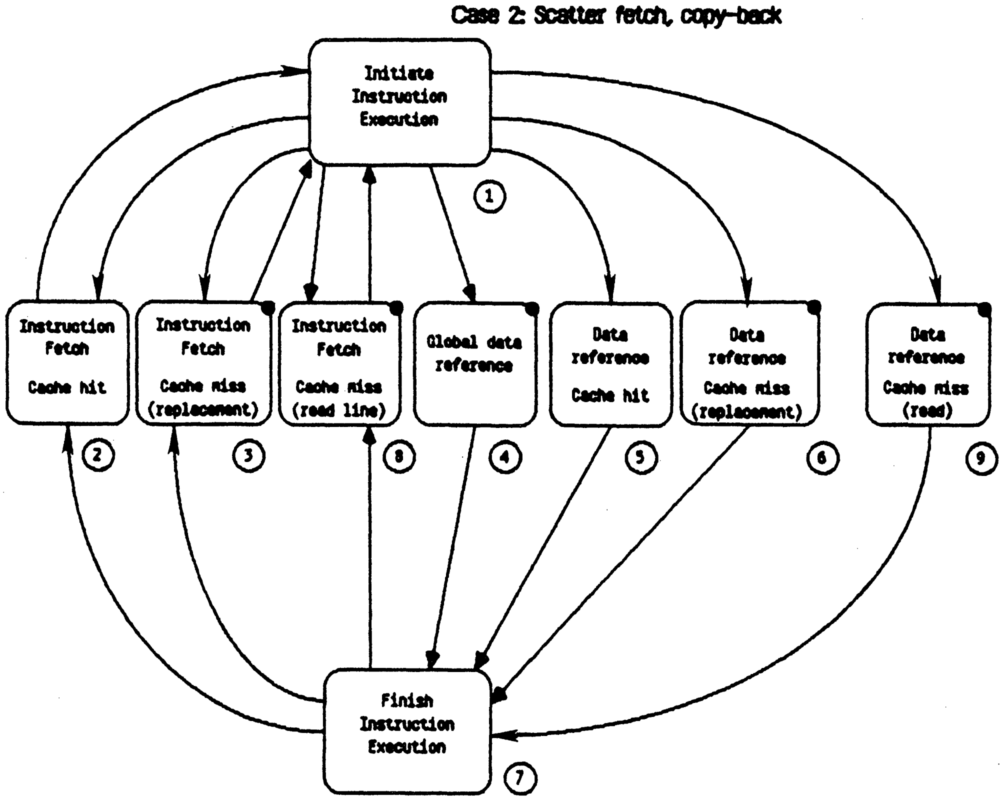

Case 2: Scatter fetch, copy-back
Initiate Instruction Execution
1
Instruction Fetch Cache hit
2
Instruction Fetch Cache miss (replacement)
3
Instruction Fetch Cache miss (read line)
8
Global data reference
4
Data reference Cache hit
5
Data reference Cache miss (replacement)
6
Data reference Cache miss (read)
9
Finish Instruction Execution
7

Figure 7.2 (continued)

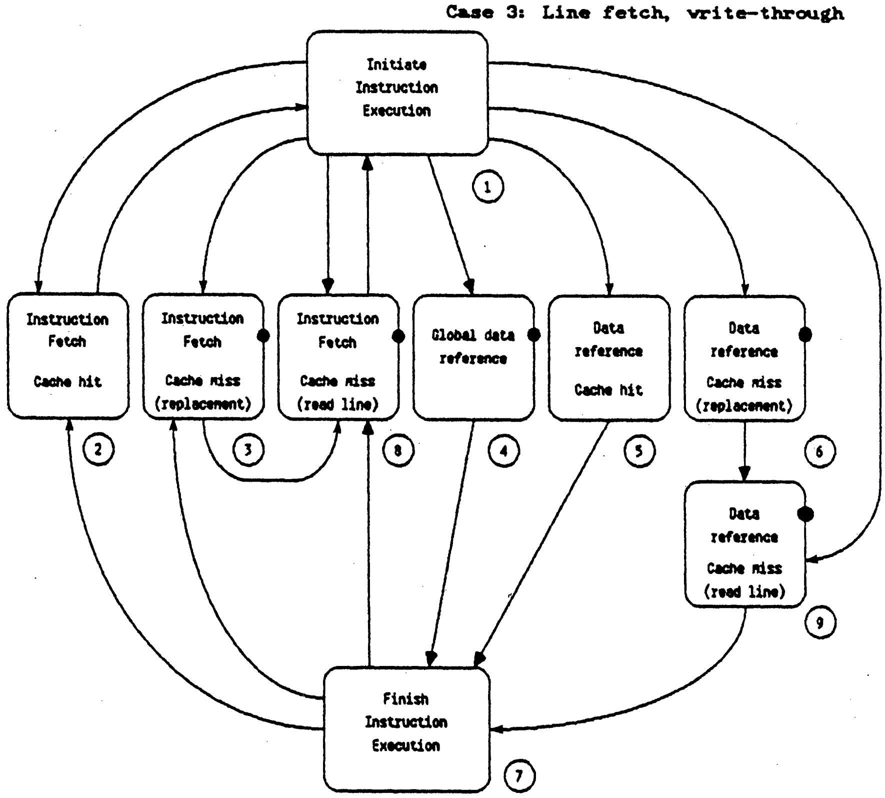

Case 3: Line fetch, write-through
Initiate Instruction Execution
1
Instruction Fetch Cache hit
2
Instruction Fetch Cache miss (replacement)
3
Instruction Fetch Cache miss (read line)
8
Global data reference
4
Data reference Cache hit
5
Data reference Cache miss (replacement)
6
Data reference Cache miss (read line)
9
Finish Instruction Execution
7

Figure 7.2 (continued)

state diagram. A different instruction execution diagram for each basic type of instruction available could be drawn and weightings could be assigned to each branch (from an instruction fetch state for example) according to their relative occurrence frequencies.

The required statistics used for instruction execution modeling might be compiled using appropriate benchmark programs executed on a uniprocessor. See figure 7.5.

7.2. Parameters of the PE Activity Model

PE state diagrams are shown in figures 7.2 and 7.3 for the two possible replacement policies considered here: write-through; and copy-back. The mean system instruction execution rate is given by:

$$\text{System instruction execution rate} = \frac{P}{\overline{C}_{p11}}$$

The parameters are:

h_I = instruction hit ratio

h_D = data hit ratio for cacheable data items

r_I = instruction reference ratio, i.e.,
the fraction of processor memory references
used to fetch instructions

r_D = data reference ratio, i.e.,
the fraction of memory references used to access data
$=1 - r_I$

r_f = the replacement fraction, i.e.,
the fraction of cache writes that cause
a copy-back to occur

f_{RO} = fraction of data references that are reads

Furthermore, divide r_D into two distinguishable subfractions r_{D_G} and r_{D_C}:

r_{D_G} = fraction of data references that pertain to noncacheable (global) data items
r_{D_C} = fraction of data references that pertain to cacheable data items
r_D = $r_{D_G} + r_{D_C}$

The above fractions will be used as probabilities, this is similar to the use of instruction occurrence frequencies as probabilities.

Again, two replacement policies will be used: write-through, where upon writing into cache, the memory management unit also writes into the appropriate location in main memory; and copy-back, where lines are copied to global memory if they need to be replaced and have been modified since being read into cache.

Therefore, four cases arise:

Case 1 - Scatter fetching with write-though replacement.

Case 2 - Scatter fetching with copy-back replacement.

Case 3 - Line fetching with write-through replacement.

Case 4 - Line fetching with copy-back replacement.

Scatter fetching is easily modeled accurately if *all* global memory references are based on a line reference. (This may be realistic if hardware only allows this type of data transfer.) That is, each time a GRM reference is made, a request is emitted *simultaneously* for all GRM's. Simultaneous emission of requests for all GRM's ensures GRM queue length equality for *all* points in time. This allows a reduction to be done in that the GRM system may be logically replaced by a single GRM.[1] Hence, in the model domain, a P × 1 system is used if $M = L$.

In the line fetching cases, the memory management unit emits a request for one of M GRM's to obtain a connection to transfer either: one or two lines of words when line transfers are required; or a single memory word move for a global data word transfer or write-through operation.

7.3. Transition Matrices and Bounds

Transition matrices and bounding schemes are detailed for the four cases. Sojourn and connection times are given by hardware timing specifications and the line length L. The cases are described next:

[1] If a single GRM request is emitted when a single data item is moved, then using the scatter fetch assumption provides an upper (depending on the fraction of single references to total references the bound may be close to the solution value) bound on mean instruction execution time. This approximate upper bound will be discussed hereafter.

Case 1 - Scatter fetching with write-through replacement.

$$\mathbf{P}_p = \left| \begin{array}{cccccccc} 0 & r_I h_I & r_I(1-h_I) & (1-r_I)(1-r_{D_C}) & (1-r_I)f_{RO}\, r_{D_C} h_D & (1-r_I)(1-f_{RO})r_{D_C} & 0 & (1-r_I)f_{RO}\, r_{D_C}(1-h_D) \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & h_I & 1-h_I & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{array} \right|$$

Within the transition matrices, products of probabilities or their complements may be interpreted as joint probabilities so that $r_I h_I$ = Pr(an instruction reference is made and a cache hit occurs). The derivation of transition probabilities is relatively straightforward.

When computing the mean instruction execution time bounds may be ascertained in a straightforward manner, whereas accurate predictions are more difficult to obtain. Since this is only an example and is not intended as a complete cache model, only bounds will be described. An upper bound on mean instruction execution time may be computed by assuming states 4 and 6 to be GRM reference states: $\Omega_{R_p} = \{3, 4, 6, 8\}$, see figure 7.2. This provides an upper bound in that it assumes a scatter (line) write occurs on write-through whereas in reality a single GRM may be singled out for the word write-through. By assuming a scatter fetch instead, the model overestimates the amount of GRM reference activity so queueing effects will be overestimated. Hence the mean state 1 cycle time will be overestimated.

Lower bounds on the mean instruction execution time may be determined in two cases: when a write-through operation delays processor activity (no concurrency within instruction execution); and when write-through takes place concurrently with processor operation. In the sequential write-through case, assume states 4 and 6 to be computation states with sojourn times given by their connection times. This provides a lower bound on the mean instruction time in that in reality more queueing will be evident than the assumptions describe and queueing time will be present in states 4 and 6 sojourn times: $\Omega_{R_p} = \{3, 8\}$.

In the concurrent write-through case assume states 4 and 6 to be computation states with no sojourn time (they are essentially removed). This provides an approximate lower bound on the mean instruction execution time.

Bounds may be derived from qualitative concepts, the SMP model's framework allows the system evaluator to reason at the SMP level. It would be difficult to do the same straightforward, qualitative reasoning when using an ARP based model [Hoo77, MuM82a].

Case 2 - Scatter fetching with copy-back replacement.

$$\mathbf{P}_p =$$

$$\begin{vmatrix}
0 & r_I h_I & r_I(1-h_I)r_f & (1-r_I)(1-r_{D_C}) & (1-r_I)r_{D_C} h_D & (1-r_I)(1-h_D)r_{D_C} r_f & 0 & r_I(1-h_I)(1-r_f) & (1-r_I)(1-h_D)r_{D_C}(1-r_f) \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & h_I & (1-h_I)r_f & (1-h_I)(1-r_f) & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0
\end{vmatrix}$$

As in case 1, upper and lower bounds on mean instruction execution times may be determined as follows: an upper bound on the mean instruction execution time is determined when state 4 is taken to be a reference state: $\Omega_{R_p} = \{3, 4, 6, 8, 9\}$. A lower bound on the mean instruction execution time is determined when state 4 is taken to be a computation state with sojourn time given by its connection time: $\Omega_{R_p} = \{3, 6, 8, 9\}$.

In both cases, states 3 and 6 represent line replacement states where a cache line is written from cache into global memory and its successor is read from global memory into cache. Hence these states' connection times are determined by the time required to write a cache line and then read a cache line (recall that it takes a single word transfer time for either operation because this is the scatter fetch case, a line is transferred in parallel to/from global memory) hence a two word transfer time would be appropriate. States 8 and 9 are entered when no replacement needs to be done, only a new line needs to be read so the connection time for these two states is given by a single line read time.

Case 3 - Line fetching with write-through replacement.

This has the same transition matrix as case 1.

Each PE request's destination is uniformly distributed over all GRM's under the reasonable assumption of interleaved address mapping into GRM numbers. Furthermore connection times for GRM references are given by the time required to read or write a line of L words from/to the chosen GRM in a circuit switched connection. The connection time for states 4 and 6 is given by the time required to move a single word between the PE and the required GRM: $\Omega_{R_f} = \{3, 4, 6, 8\}$.

Again, by taking the write-through states (4 and 6) to be computation states of negligible sojourn time an approximate lower bound on the mean instruction execution time may be obtained.

Case 4 - Line fetching with copy-back replacement.

This case is similar to case 2 but states 3 and 6 must be separated to represent the two possibilities for request emissions: if, on a line replacement, both the line to be written and the line to be read fall into the same GRM (with probability $\frac{1}{M} \frac{1}{M} = \frac{1}{M^2}$), then a single connection transfers $2L$ words; alternatively, if the two lines to be moved fall into different GRM's (with probability $1 - \frac{1}{M^2}$), two references are made. If the two references are made sequentially (due to lookahead unit complexity limitations), an appropriate state diagram is given in figure 7.3.

The transition matrix is similar to case 2 so will not be displayed, modifications in transitions relevant to entries to states 3 and 10, and 6 and 12 are required. Notice that the state 12, 13 sequence describes the emission of two requests for line transfers and as such must in actuality reference two distinct GRM's. This could be represented by separating these two states into M similar parallel copies as shown in figure 7.4 -- one for each GRM referenced on the first request emission. The same applies to the state 10, 11 sequence. To reduce complexity the parallel states in figure 7.4 might be lumped (with some error ensuing) as in figure 7.3 with $\eta_{p,12,m} = 1/M$, $\eta_{p,13,m} = 1/M$. Recognize that this is an approximation. The model is powerful enough to be used more accurately by using the M parallel cases substituted for the approximate states 12 and 13 (the computed results may though be the same for the explicit and lumped configurations because of the fractional argument). Again concurrent processor operations may be examined to obtain bounds on the mean instruction execution time.

If cache write and reads occur concurrently when distinct GRM's are referenced, a PE may emit *two* requests simultaneously. This violates the single request per PE assumption so the SMP model as described so far may be difficult to apply. The next chapter discusses SMP model extensions that may be used to model multiple request emissions, the problem has not generally been solved sufficiently.

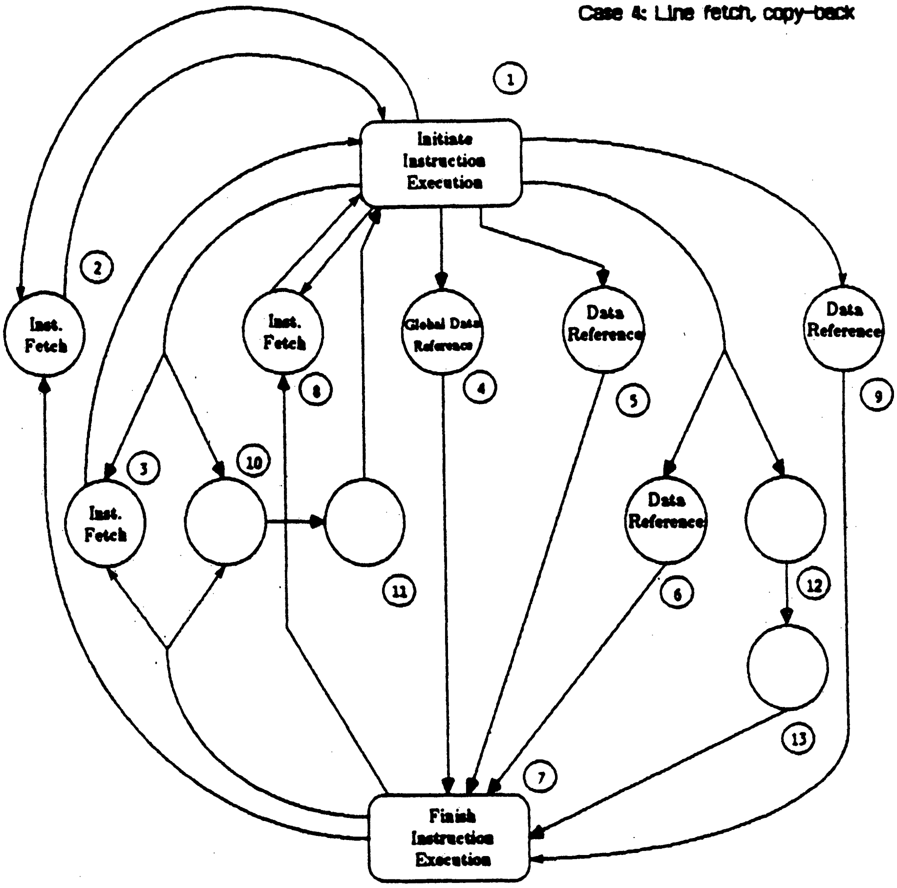


Figure 7.3 Case 4 instruction execution timing.

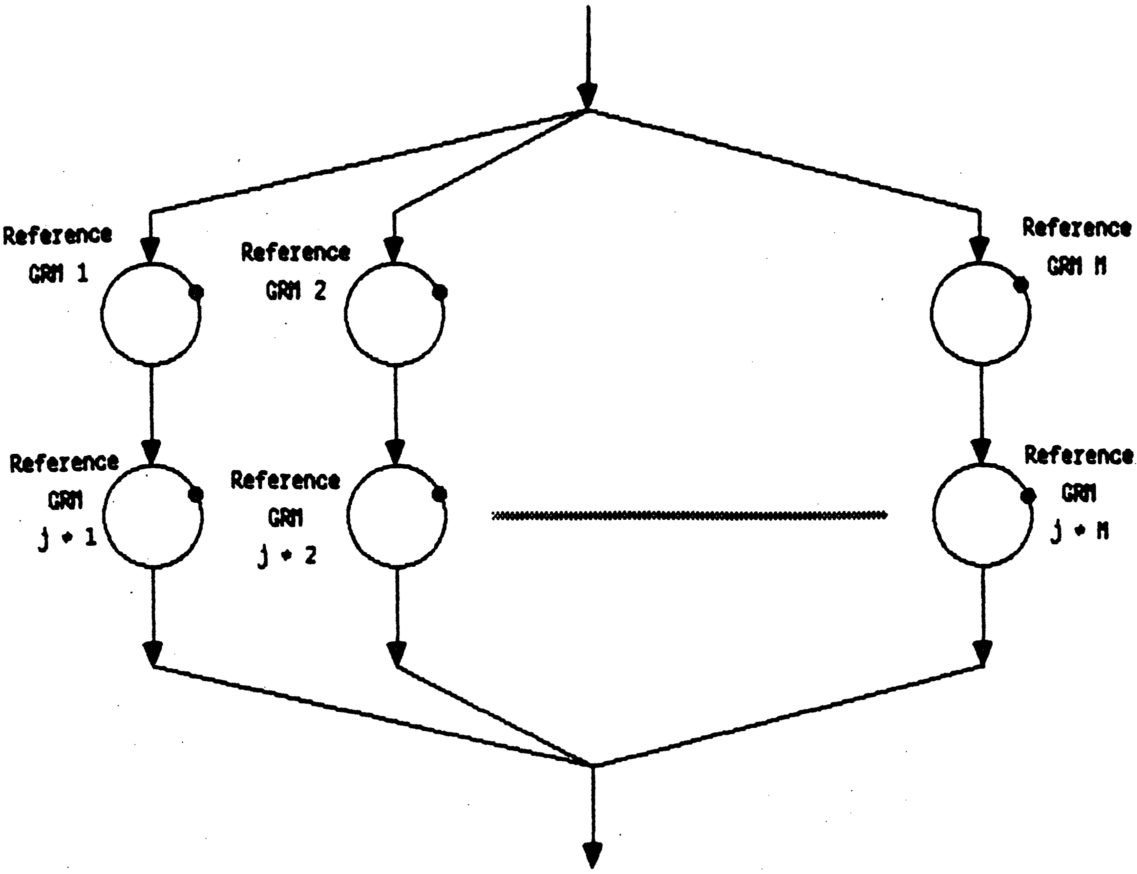


Figure 7.4 *M* parallel reference states to replace states 12, 13 and 10, 11.

7.4. Conclusions

This chapter demonstrated an application for the SMP model in a design comparison study. The example shows the natural descriptions that may be used to represent PE behavior at a low level of PE operation. Had the ARP based models of processor behavior been employed, little could have been said about instruction execution times. For example, using ARP based models requires either: inter-reference and connection times be determined experimentally (on a uniprocessor for example), instruction execution times are still undeterminable from the ARP level of description; or an SMP to ARP reduction could be done (to coerce the cache state diagrams into an ARP model description), then waiting times could be found from the chosen ARP model, and finally an SMP description could be used to derive instruction execution times from waiting times. In short, if timing measures are desired then some form of SMP analysis will be required. The SMP model supplies directly these aspects of timing analysis.

Clearly, the instruction model described here is a simplification of processor timing evident in real systems. The description may be extended to include more realistic timing descriptions. Figure 7.5 shows a more sophisticated instruction execution diagram where various types of instructions have been separated into different branches off of an instruction fetch state.

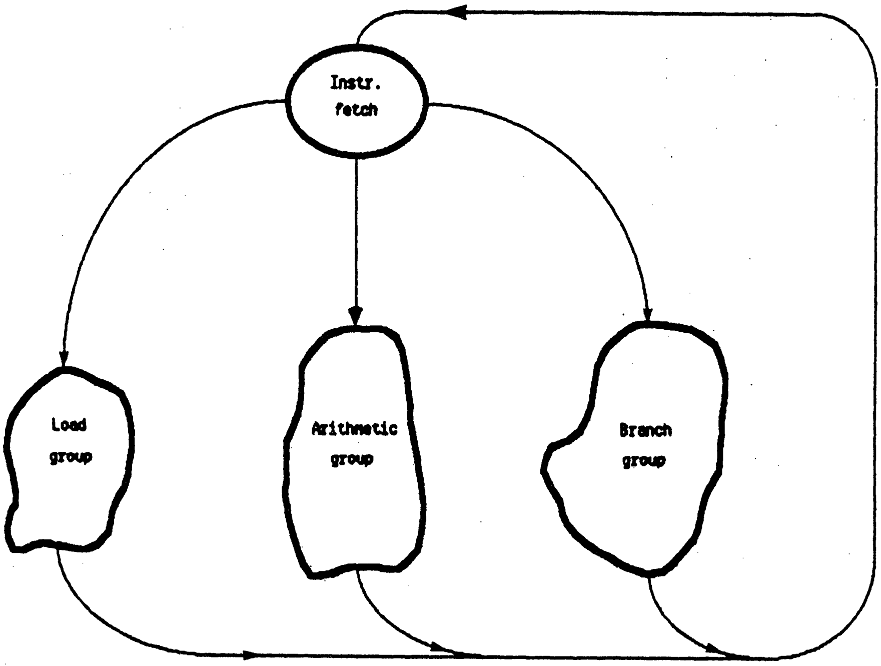


Figure 7.5 Simple extension of the instruction execution model.

CHAPTER 8

MODEL EXTENSIONS AND MODIFICATIONS

Model extensions are described in this chapter. Statements or calculations are made when appropriate. The emphasis is on describing problems that have been encountered but not yet solved sufficiently. Hence further work might address the model modifications described.

8.1. State Occupation Overlapping

Since processor hardware often overlaps operations (e.g., lookahead, etc.) modeling such behavior is desirable. Figure 8.1 shows an example where a PE performs two operations concurrently in a fork/join operation. To model this behavior, concurrently occupied states may be lumped together into a single composite state. The sojourn time for the composite state is required by the SMP model. Clearly, the sojourn time for composite states is given by:

$$\tilde{S}_{pC} = \max_{s \in C}\{\tilde{S}_{ps}\}$$

Where C is the "composite" state composed of a set of states $s \in C \subset A_p^r$. Further exact calculations for moments of composite state sojourn times *require the CDF's* $S_{ps}(t)$. The CDF's are required because

$$\bar{S}_{pC} = \int_0^\infty (1 - S_{pC}(t))dt$$

$$S_{pC}(t) = Pr(\tilde{S}_{pC} \leq t) = Pr(\tilde{S}_{ps} \leq t, \textit{for all } s \in C)$$

If all states $s \in C$ have independent sojourn times this becomes

$$S_{pC}(t) = \prod_{s \in C} S_{ps}(t)$$

Bounds may be placed on $\bar{S}_{pC}$ without $S_{ps}(t)$ though:

$$S_{pC}(t) \leq S_{ps}(t) \;\; \textit{for all } s \in C, \textit{ and } t \in [0,\infty)$$

because the above product is less than any term in the product. Hence,

$$1 - S_{pC}(t) \geq 1 - S_{ps}(t) \;\; \textit{for all } s \in C, \textit{ and } t \in [0,\infty)$$

$$\overline{S}_{pC} = \int_0^{\infty}(1-S_{pC}(t))dt \geq \int_0^{\infty}(1-S_{ps}(t))dt = \overline{S}_{ps} \;\; \textit{for all } s \in C$$

So,

$$\overline{S}_{pC} \geq \max_{s \in C}\{\overline{S}_{ps}\}$$

Taking this bound to be an equality would give a lower bound on cycle times (state transition times) -- assuming that $\overline{W}_{pm}$ are known -- in that composite states may be occupied for more time than ascertained by the bound. Computationally, taking the above bound to be an equality is unpredictable. Consider assuming the bound to be an

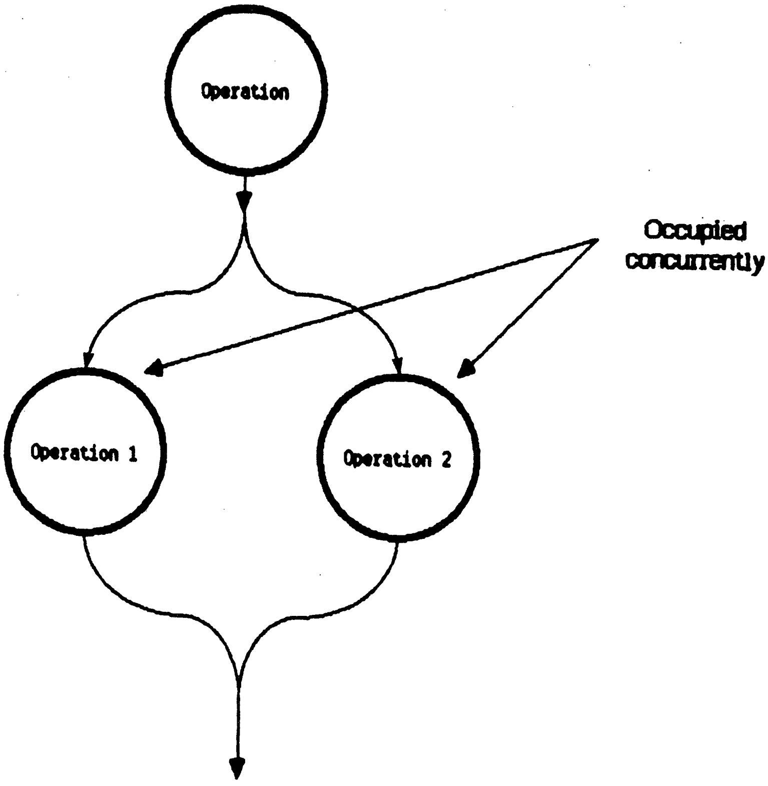


Figure 8.1 Overlapped state occupation.

equality, and consider the solution process on the k^{th} iteration of the fixed point solution procedure. Due to equation (3.24), assuming the lower bound makes computed rates on the k^{th} iteration higher than actual rates. These rates then raise waiting times computed on the $k+1^{st}$ iteration, which in turn increases the assumed lower bound for reference state sojourn times and decreases rates computed on the $k+1^{st}$ iteration. This then reduces the waiting times computed on the $k+2^{nd}$ iteration. Hence as iterations proceed, $\bar{S}_{pC}$ may not be necessarily be computed as $\max\{\bar{S}_{ps}\}$, the computed value oscillates initially but will (probably) converge to a particular value of unknown quality. Although approximate results may be obtained using this technique, their quality has not been verified. Perhaps further work is warranted on this aspect. If composite states are defined, it will be necessary to modify previous notation for request probabilities, etc. These aspects have not been pursued or formalized.

To model concurrent PE activity during GRM references (such as when processors overlap computations with resource use) other techniques may be developed. For example, when overlapped references and computations are to be modeled the desired model property is to have GRM reference times (for overlapped states) not affect SMP transition time characteristics; that is, the sojourn time for overlapped reference states should be zero, they should not delay PE state machine sequencing. The technique will be to coerce the overlapped states into the context of the present SMP model. This may be done as follows: associate with appropriate reference states (whose sojourn times are to be neglected) "anti-reference" states.

Consider a reference state and its anti-reference state: the anti-reference state is occupied just prior to the overlapped reference state so that the two (reference and anti-reference states) are always paired together. (When the anti-reference state is exited, its associated normal reference state is entered with probability one.) Furthermore, the anti-reference states' sojourn time is given by the negative of the its associated normal reference state.. No requests are emitted upon entries to anti-reference states, a request is emitted at the entry to the normal reference state (the converse choice is also valid). Since the two states are occupied sequentially, the net delay encountered (in the model, as viewed from the PE/GRM interface) during the resource use is zero, but a request *is* emitted. (State occupation ordering could be reversed, the sequence is irrelevant.) Hence GRM queueing times will be affected as they should be. Consider a stochastic path,

$\tilde{Z}_p(t)$: a transition is made *through* the overlapped reference state instantly and PE state machine sequencing proceeds.

For example, consider a simple PE description where a computation state (of sojourn time 2 say) leads to a reference state (with connection time 1 say) which is occupied "concurrently" with the computation state so that the time between request emissions is 2 units (the computation state cycle time). The situation may be depicted as in figure 8.2 with the appropriate anti-reference state. There are still unsolved problems with this technique. If PE's emit requests faster than the global resource system may service them (the effect might be ignored if it occurs with small probability) a resource system "overflow" condition will occur. This would be the case if a system has $P >> M$ so that $\overline{W} > 1$.

Care must be used in applying the previous notation to anti-states it is not appropriate to ask about the fraction of time that a PE spends in anti-reference states, for this would be negative. It seems that statements may only be made about anti-reference states paired with their corresponding normal states. If the sum is used for finding the fraction of time that a PE spends in a reference/anti-reference state pair and (3.12) is used for each state (reference and anti-reference), then the result is zero as it should be -- the PE spends no measurable time in overlapped reference states.

In summary, the SMP model may be enhanced with this state type. The utility of anti-reference states suggests that there may be a use for anti-computation states. Their existence and use have not been examined.

8.2. Multiple Requests

Consider extending the SMP model framework to allow PE's to emit multiple requests. For example, lookahead units may emit multiple requests before connections are required. There are, though, problems involved with multiple request emissions aside from the obvious data dependency requirements.

Consider an example where two requests are emitted simultaneously for different GRM's (i and j) when a reference state is entered by a given PE. The two requests are placed in GRM queues i and j until the connections may begin. If the connection to GRM i becomes available first, PE interface hardware starts the circuit switched

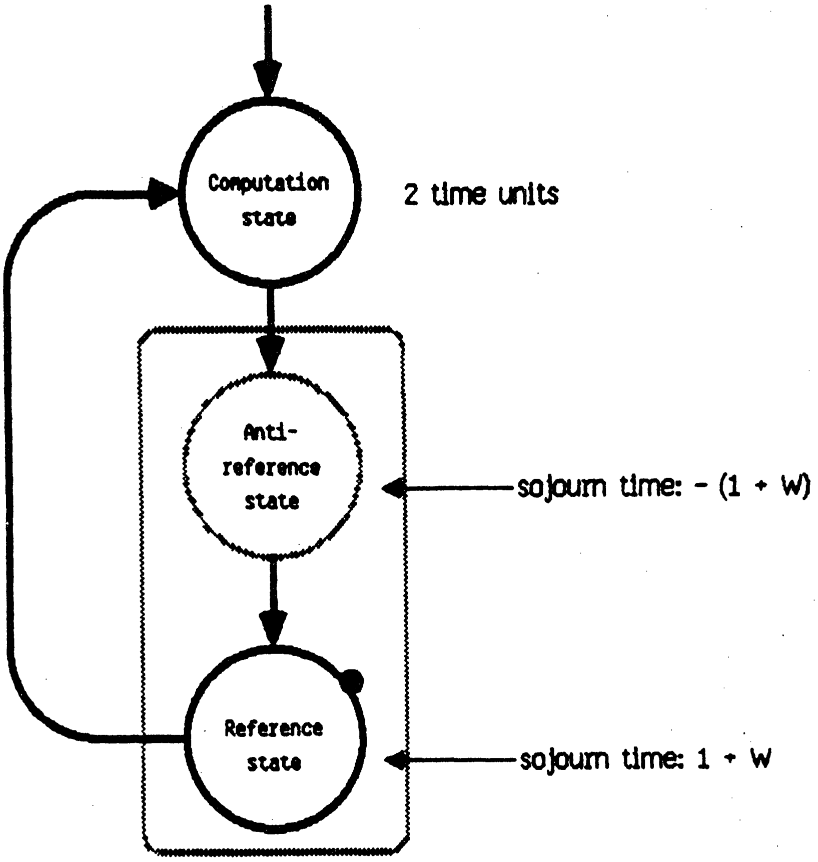


Figure 8.2 Anti-reference state example.

connection between the PE and GRM *i*. If during the GRM *i* connection the GRM *j* connection becomes available, GRM *j* must wait until the PE may use it. This in effect places a return queue for connections at each PE. Notice that GRM *j*, while waiting for the required PE, may either: continue waiting, in which case effective connection times have been increased at the expense of waiting times; or it may temporarily activate another PE's connection if there is one in queue. The second choice seems to introduce much more complexity into the design of GRM's, so much so that it is doubtful that preemptive connections would be used at the PE/GRM level of implementation, except with very slow devices such as disks. The first choice is most appropriate but again will effectively increase waiting times. In any event, it may be necessary to ensure temporal integrity in returning connections due to possible data dependencies.

It is believed that the first case might be modeled by enhancing the SMP model roughly as follows: new notation for request emission probabilities (η_{psm}) must be devised. For example, rather than

$$\eta_{psm} = Pr(\textit{PE p emits a GRM request when it enters state s and it is for GRM m})$$

we might define

$$\epsilon_{psm} = Pr(\textit{PE p emits a request for GRM m when it enters state s}).$$

Then $\epsilon_{ps1} = 1$, $\epsilon_{ps2} = 1$, etc. may be allowed. The mean number of requests emitted in state s would be $\sum_m \epsilon_{psm}$.

For reference states, multiple simultaneous emissions may be modeled as concurrently occupied states, see figure 8.3. Due to analysis complexity, it is believed that several moments of waiting and connection times (including the effects due to return queueing also) will be required. Formulations including CDF's are possible, they have not been pursued.

An aspect related to concurrent and overlapped timing is the concept of optimal lookahead (prefetch) operation. Lookahead request emission may be used to overcome degradation attributable to waiting times. Consider a simple single request per PE case and an ARP description of PE behavior as in figure 8.4. Here PE p emits a request $\overline{W}_p = \sum_m \overline{W}_{pm}\, \eta'_{pm}$ units of time before the connection is required. This allows the *mean* rate of ARP cycling to achieve its potential (where queueing time is zero). Note that *the effects from lookahead timing themselves must be included in the analysis* in that too long a lookahead will yield idle GRM's because connections will be established too soon, PE's will not be ready to use the GRM's at the time the connections are ready. Alternatively, if the lookahead unit underestimates queueing times, then a mean waiting time will still be present. It does not suffice to compute $\overline{W}_p$ independent of the lookahead unit and then set the lookahead time to this value. In the ARP domain of PE description, a technique for finding this optimal lookahead time is to set the computation state sojourn time with lookahead ($\overline{S}_{p1}''$) to $\overline{S}_{p1}' - \overline{W}_p$. That is, in each iteration of equation solution set $\overline{S}_{p1}'' = \overline{S}_{p1}' - \overline{W}_p$ and use $\overline{S}_{p1}''$ as the equivalent state 1 mean sojourn time. The final solution that results for $\overline{W}_p$ represents the lookahead time that is appropriate.

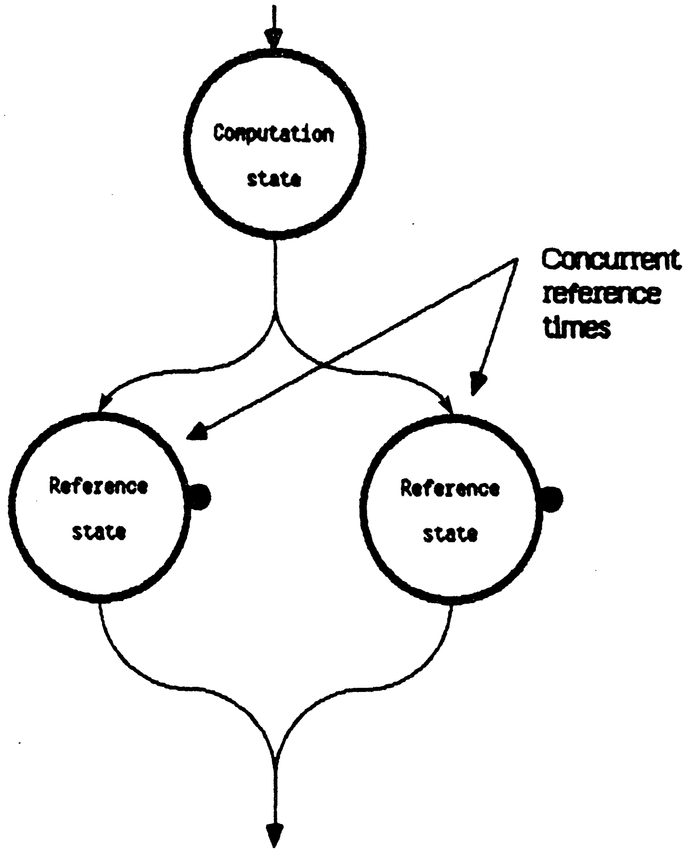


Figure 8.3 Multiple request timing example.

If $\overline{W}_p = \sum_m \overline{W}_{pm} \eta'_{pm}$ is chosen as the lookahead for all GRM's, it might not be appropriate for a particular GRM, i.e. $\overline{W}_{pm}$ may not be $\overline{W}_p$ except in symmetric situations.

It is possible that even in the symmetric situation $\overline{W}_p > \overline{S}'_{p1}$ in which case no optimal (in the sense of processors seeing no queueing time) solution exists for the lookahead time.

Since the SMP to ARP reduction is involved in determining lookahead times and it ignores the dependence of waiting times on emission states, error ensues in that the ARP description does not guarantee instantaneous temporal equivalence. That is, if two GRM references occur sequentially and rapidly, the lookahead unit -- it uses $\overline{W}_p$ as its lookahead time -- may not compensate for each reference queueing time. The deception arises from the use of the ARP in describing processes which are not actually renewal processes -- time varying behavior has been approximated with steady-state statistics, and average values are used as the lookahead times.

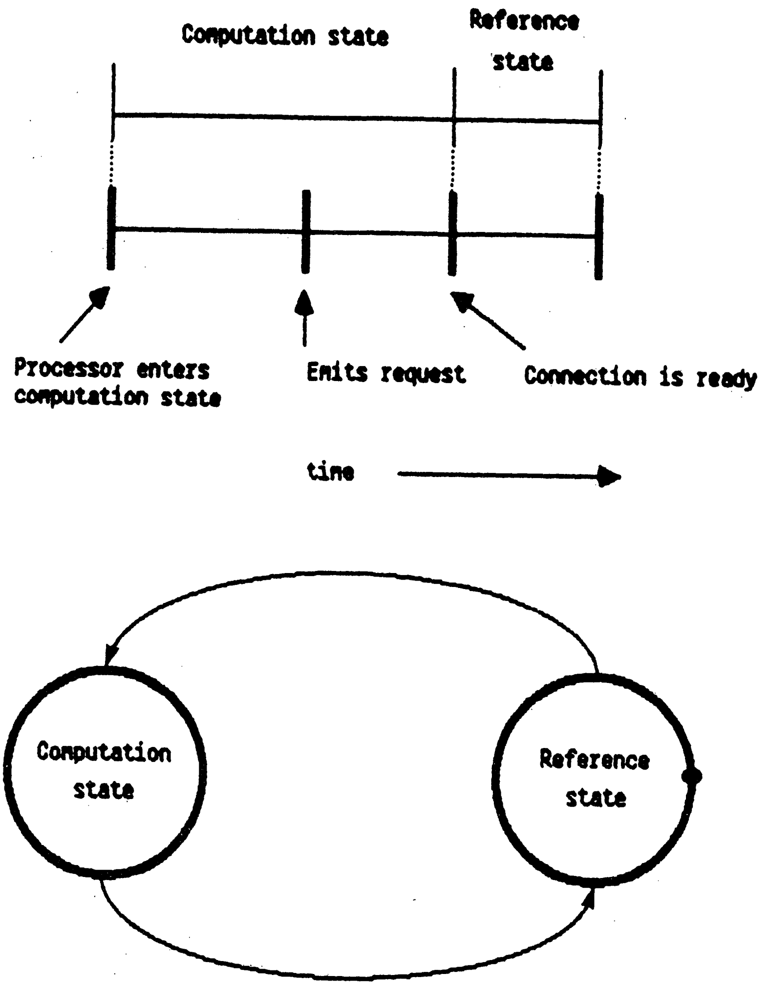


Figure 8.4 Lookahead timing.

8.3. Operating System Effects

The basic SMP model has assumed that a single process executes on each PE. Meaningful results are believed to be obtainable in multiprogrammed situations if job mixes are known for each PE and detailed information on operating system behavior is available.

Again, the key to finding an individual program's mean execution time (CPU time, not elapsed user time, this is dependent on scheduling algorithms, etc.) is the prediction of mean queueing times. Mean queueing times may be estimated by using an average mix of "jobs" that execute on PE's. Effects due to PE i on PE j would be weighted by the fraction of CPU time received by various tasks/programs executing on PE i, including the operating system running on PE i.

Although the approach may seem complex, it is simply an approach whereby PE behavior is modeled as the average behavior of those tasks that execute on the PE. Lumping programs together that exhibit much different resource use in various phases of execution may cause an accuracy problem to arise. Consider an example, suppose we are interested in computing waiting times for PE 2's program in a dual processor. During program 2's execution however, PE 1 switches context from one program to another. After the PE 1 context switch, PE 2 waiting times may change drastically, depending on the characteristics of the two PE 1 programs. The effects may not be predicted properly with simple fractional mixes.

8.4. Phases of Computations

When programs execute in phases, or PE's switch context, the problem of system behavior with these phase changes must be addressed. The complexity of phased analysis is large if done rigorously and it seems that simplifying approximation must be employed. An example of a possible modeling technique is the construction of a system wide SMP whose states represent the set of programs executing on each PE at the given time. Transitions among these global SMP states correspond to context changes in PE's, see [MaM82]. This approach appears to be intractably complex.

Programs which execute in phases might be modeled by abstracting the SMP model equations to a higher level where states in the new high level SMP model are in themselves SMP's of the basic variety described thus far. The abstraction provides a straightforward technique for accommodating phased computations. This seems to be a reasonable, tractable approach to phase analysis.

8.5. Process Communication

Consider modeling various forms of process/program communication where programs executing on different PE's communicate with each other. Two forms of process communication will be discussed: direct process communications where PE's (their state diagrams) exhibit direct state occupation coupling; and indirect process communication through message queues contained in global memory.

8.5.1. Direct Process Communication

The simplest example of two processes communicating directly where a sender process writes a message into, for example, a mailbox location. A receiver process checks its mailbox to see when the message arrives. In the case where the sender waits for the receiver to "catch up," a process join has been created (the receiver waits for a message if it checks the mailbox first), see figure 8.4 -- this is the case considered in this subsection. (The other situation, where a message queue may form, is considered next.) Define the communication states for processes i and j to be s_i and s_j respectively. Then from the connectedness of process behavior:

$$\tilde{C}_{is_i s_i}(\omega) = \tilde{C}_{js_j s_j}(\omega) \tag{8.1}$$

where ω represents the particular process cycle (from s_i to s_i) in question. This reflects the fact that neither process may get ahead of the other in terms of cycle times with respect to their communication states.

Consider the sojourn time for each communication state s_i and s_j. If these may be established, then as before, each SMP will have been sufficiently characterized. Keep in mind that the concepts certainly need to be extended to the case where are multiple communication states in many SMP's.

From the characterization of the join above, note that one and only one of the two processes will wait in its communication state for a nonzero time, hence:

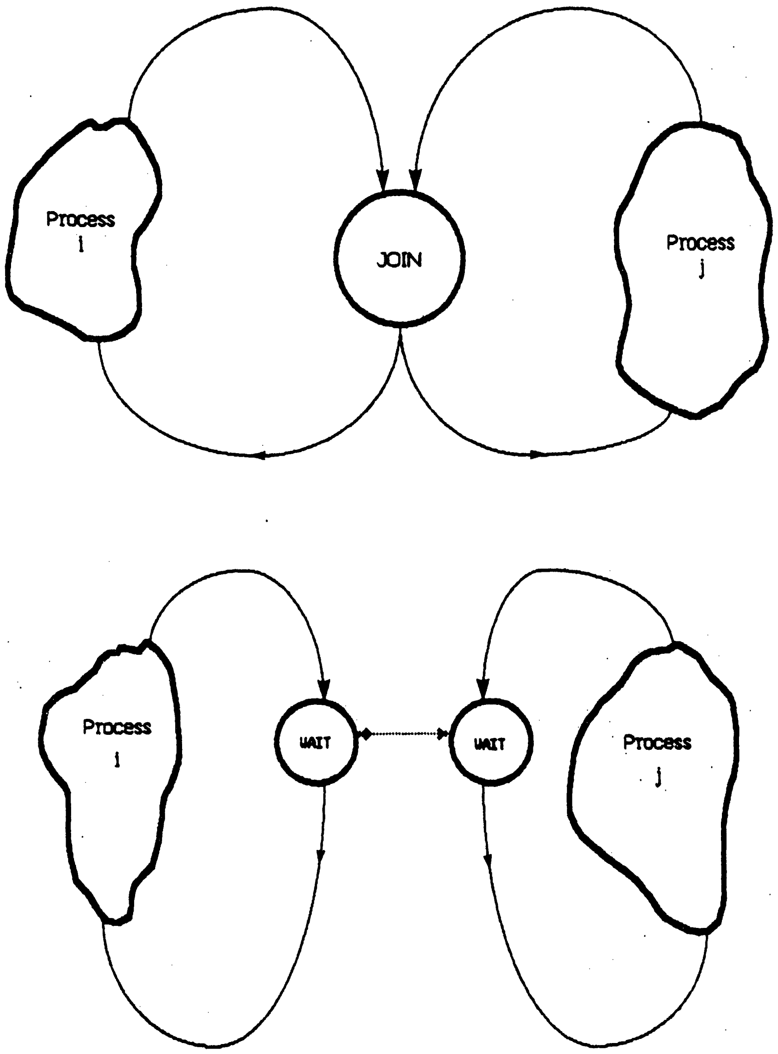


Figure 8.5 Direct communication process linking.

$$\tilde{S}_{is_i}(\omega) = 0,\ \tilde{S}_{js_j}(\omega) > 0 \ \ or \ \ \tilde{S}_{is_i}(\omega) > 0,\ \tilde{S}_{js_j}(\omega) = 0$$

or,

$$\tilde{S}_{is_i}(\omega) \times \tilde{S}_{js_j}(\omega) = 0 \qquad (8.2)$$

The problem is that neither (8.1) or (8.2) uniquely identifies mean cycle times or sojourn times from the solution point of view. (8.1) establishes the equality, but does not describe how a solution may be obtained. (8.2) is informative, but is of little use in establishing either mean value (the two random variable $\tilde{S}_{is_i}(\omega)$ and $\tilde{S}_{js_j}(\omega)$ are *not* independent). It appears that the only approach to finding $\overline{S}_{is_i}$ or $\overline{S}_{js_j}$ and consequently $\overline{C}_{is_i s_i} = \overline{C}_{js_j s_j}$ is the computation of approximate CDF's for message waiting times. For example, the mean sojourn times may be characterized as follows

$$\overline{S}_{is_i}(\omega) = E[\ \textit{message waiting time} \mid \textit{process } i \textit{ must wait}\ ]Pr(\textit{process } i \textit{ must wait})$$

$$Pr(\textit{process } i \textit{ must wait}) = Pr(\tilde{T}_i(s_i,\{s_i\}) < \tilde{T}_j(s_j,\{s_j\}))$$

$$= 1 - Pr(\tilde{T}_i(s_i,\{s_i\}) \geq \tilde{T}_j(s_j,\{s_j\}))$$

from the previous notation. The *only* way to compute this seems to be as follows (assuming that both PE's i and j leave their states s_i and s_j simultaneously):

$$Pr(\textit{process } i \textit{ must wait}) = 1 - \int_0^\infty T_j(s_j,\{s_j\},t)dT_i(s_i,\{s_i\},t)$$

Which requires the state transition time CDF's (or at least cycle time CDF's). Representative functions might be assumed for $T_j(s_j,\{s_j\},t)$ and $T_i(s_i,\{s_i\},t)$ from which moment matching may be used to find various unknowns in the representative functions. The form of these representative functions should be determined from real program communication data.

Due to (8.2):

$$Pr(\textit{process } i \textit{ must wait}) = Pr(\tilde{S}_{is_i} > 0) = Pr(\tilde{S}_{js_j} = 0).$$

To compute the mean time that process i must spend waiting for process j, we might consider:

process i message waiting time (ω)

$$= \begin{cases} \tilde{T}_j(s_j,\{s_j\})(\omega) - \tilde{T}_i(s_i,\{s_i\})(\omega) & \tilde{T}_j(s_j,\{s_j\})(\omega) > \tilde{T}_i(s_i,\{s_i\})(\omega) \\ 0 & \tilde{T}_i(s_j,\{s_j\})(\omega) \leq \tilde{T}_i(s_i,\{s_i\})(\omega) \end{cases}$$

That is, we are finding the conditional expectation of the difference between entry times to the communication state. The calculations appropriate are (abbreviate $\tilde{T}_j = \tilde{T}_j(s_j,\{s_j\})(\omega)$, $\tilde{T}_i = \tilde{T}_i(s_i,\{s_i\})(\omega)$):

$$Pr(\textit{process } i \textit{ message waiting time} \leq \alpha \mid \textit{process } i \textit{ must wait}) =$$

$$
\begin{aligned}
Pr(0 \leq \tilde{T}_j - \tilde{T}_i \leq \alpha) & \\
&= Pr(\tilde{T}_i \leq \tilde{T}_j \leq \alpha + \tilde{T}_i) \\
&= \int_0^\infty Pr(\beta \leq \tilde{T}_j \leq \alpha + \beta) dT_i(\beta) \\
&= \int_0^\infty (Pr(\tilde{T}_j \leq \alpha + \beta) - Pr(\tilde{T}_j < \beta)) dT_i(\beta) \\
&= \int_0^\infty (T_j(\alpha + \beta) - T_j(\beta) + Pr(\tilde{T}_j = \beta)) dT_i(\beta) \\
&\equiv \mathbf{W}_i(\alpha)
\end{aligned}
$$

Then the conditional waiting time is given by:

$$E[\textit{process } i \textit{ message waiting time} \mid \textit{process } i \textit{ must wait}] = \int_0^\infty (1 - \mathbf{W}_i(t))dt$$

To perform these computations tractably, representative functions might be used. Also note a very important characteristic which affects an accurate analysis: since processes i and j exit their communication states simultaneously there is strong correlation/dependence between states occupied in processes i and j. It may not be appropriate to use general time quantities when computing effects on process i due to process j at the level of detail required in this application. It seems, though, that to do otherwise introduces time dependence into the physical queueing analysis.

The form of message waiting used in the implementation impacts GRM queueing times. Consider a technique for message waiting where receiving PE's repeatedly check their message queues (contained in GRM's) in a tight loop (referred to as a spin lock in C.mmp terminology, [SiB82]). Here the message waiting process causes its own GRM queueing times to increase because of the high rate of request arrival (from the spin lock) at the GRM in question. Alternatively, if special "interrupt" hardware signals message readiness, then there would be no increase in GRM queueing times due to spin locks -- there would actually be a decrease because no GRM references would be made in message wait states.

Notice that since the receiving process changes its characteristics when it is waiting for messages, the problem of phase analysis is deeply embedded in the modeling of communicating PE's. Transition probabilities will also be a function of sojourn times in that the number of check loops executed by a receiving process in its wait state depends on the mean message waiting time due to the sending process and the mean queue check time (with GRM queueing times included in the spin lock case). See figure 8.6 for a simple diagram. With this addition of sojourn and queueing time dependent transition probabilities, the complexity of the solution process increases greatly. Each fixed point iteration would, besides the previous calculations, require solving linear equations for the embedded MC probabilities.

8.5.2. Indirect Process Communication

In this form of communication, processes communicate through message queues. We will associate with receiving processes message queues (which would physically be located in global memory) that senders deposit messages into. Aspects of importance in this situation include the mean message queue lengths, message queueing time moments, logical message queue server utilizations, etc. To obtain characteristics of message queue behavior, message queue service and arrival processes must be characterized. To use, for

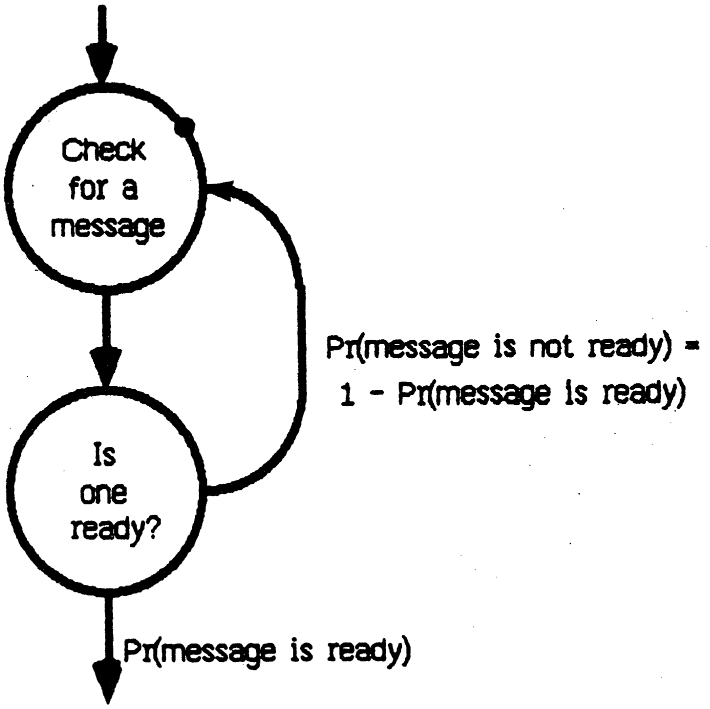


Figure 8.6 Bernoulli message waiting.

example, a two moment based message queue model requires the first two moments of message interarrival times and message queue service times. Both of these parameters may be deduced (approximately) from considering the message source and service process SMP transition times. That is, SMP timing characteristics directly affect the behavior of these logical message queues.

For example, consider a message queue that a single process (SMP i) serves. Furthermore, suppose that all other processes deposit messages into this message queue. The physical implementation (such as its storage placement) of the message queue provide details for the SMP model. Again, it is believed that the problem of phases is involved. Note that this form of communication is not wholy distinct from the direct form, a queue is simply allowed to form at the receiving process.

8.6. Conclusions

The discussion has been incomplete in many places in this chapter, it is intentional as the concepts are only meant as guidelines for further study directions. The problem of phase analysis seems to be related to many other model enhancements shown. State occupation overlapping and multiple request emission modeling also seem to be important enhancements.

CHAPTER 9

CONCLUSIONS AND SUMMARY

The model framework described is applicable to specific model development applications and provides general information about characteristics of system behavior. In its present state, the SMP model of requestor/resource systems is more powerful than previous models specifically developed for modeling memory interference. The SMP model supplies many of the necessary tools for constructing system models with the PE description level chosen by the system evaluator. PE description levels may range from the elementary alternating renewal process description considered in previous memory interference models, through renewal based descriptions such as instruction execution descriptions, up to complete program execution descriptions.

Attention to detail in the model development should help to ensure model applicability in many situations. Apart from applications in system studies, the model analysis results provide analytic insight into system behavior in a more detailed manner than previously available. The generality in development allows the model to be applied in non-computer applications including finite customer queueing network analysis.

In systems where requestors whose state transitions may be approximated by semi-Markov processes and contend for system resource module service through a virtually circuit switched crossbar, the SMP model as described may be applied. In situations other than those described here, some equations may have to be re-written to reflect specific system operation and new physical analysis may have to be done, but the entire model framework need not be re-developed. This flexibility is achieved by using physical analysis in characterizing resource system behavior. Physical analysis maintains reasonably low solution complexity even in reasonably general situations.

The SMP model of requestor activity allows requestor to be specified in a versatile, natural way; the model calculations operate on requestor description parameters. Parameters include: requestor state transition probability matrices; computation state sojourn time moments; resource module service time moments; and resource module reference probabilities, or reference patterns. The number of moments of time related random variables required by the model is determined by the physical analysis. Using first moments may allow bounds to be obtained.

With these parameters (one set applies to each requestor) the SMP model is capable of predicting the following system performance measures: requestor state transition time moments, and hence mean execution times and rates; system element utilizations and data transfer rates; resource waiting time moments; and coefficients of variation for timing quantities. The meaning of the timing performance measures is determined by the requestor activity representation chosen; that is, if processor activity representations apply to program execution directly, then program execution times are predictable from the model directly; alternatively, if processor activity representations apply to instruction execution timing, then instruction execution time moments may be predicted by the model.

Along with the prediction capability is the general information obtainable from the mathematical relationships. Summarizing some of the properties and results that have been discussed here, but not necessarily in their order of importance:

- higher moments of service and computation times may be important if requestor synchronization is to be described

- resource queueing times are asymmetric in general, not all requestors "see" the same resource system behavior

- in some system configurations (the system hardware description along with the requestor state machines) resource queueing *must* exist, there may not be a perfect synchronization scheme

- assuming resource queueing times to be requestor state invariant is approximately equivalent to the reducibility of a requestor state machine (semi-Markov process) to an alternating renewal process (of the sort considered in previous models) that is useful for finding waiting times and other quantities not related to semi-Markov process state transition times

- program execution times may be predicted reasonably accurately from model parameters

- requestor activity may be modeled in a direct and natural way, state diagrams are used to represent timing and resource reference characteristics of requestors

- the model framework is general enough that special case models may be devised for specific purposes, for example, cache models may be devised in a straightforward manner

- enhancements of the basic model allow process communication timing modeling to be done

- experimental evidence seems to show that first moment matching suffices for deducing program transition probabilities, loops may be represented probabilistically with reasonably accurate results for first moment timing predictions

- bounds on system performance measures may be derived from first moment quantities alone

- the model developed may be used for timing based comparison studies

- coefficients of variation may be computed, they provide information about system behavior randomness

- resource utilizations are a robust quantity with respect to waiting time predictions

With the chapter 8 discussion on model extensions, it is believed that useful enhancements may be made without major revisions in the ideas and relationships described. Since the SMP model allows requestor descriptions to be formulated in a natural way, it is doubtful that the model framework would need to be fully revised in developing more modeling accuracy or capabilities. Enhancements and modifications may be made as required in modeling new effects.

APPENDIX

There seems to be an error in the calculations concerning the superposition of two hypoexponential (coefficient of variation less than one) renewal processes as shown in [Kue79]. The calculations described here are guided by [Kue79].

When superimposing two renewal processes (counting processes for example) the resultant process is a renewal process only when the two original processes are. In general, the result of a superposition is not a renewal process (only if both source processes are Poisson is the result a renewal -- and Poisson -- process). Treating the resultant process as a renewal process leads to an approximate coefficient of variation. The approximation is most accurate when the renewal times for the superimposed processes have densities on an interval. The approximation is least accurate when the two source process renewal times are discrete in nature. [Kue79] proposed representative processes which are based on the first two moments of the source processes (see figure A.1). Renewal process theory will be used to compute a coefficient of variation for the resulting process. Notice that three cases ensue: both source processes have coefficient of variation less than 1; one source coefficient of variation is less than 1, the other is greater than 1; and both source coefficients of variation are greater than 1. The first case will be derived here, the second two are covered in [Kue79].

Call C the resulting coefficient of variation while C_1 and C_2 are the two source coefficients of variation. Let $\tilde{T}_V$ be the forward recurrence time (in the result process) from an arbitrary examination time for the result process. Let $t_1 = E$[renewal time for process 1], $t_2 = E$[renewal time for process 2]. Then:

$$C^2 = 2\frac{t_1+t_2}{t_1 t_2}\,\overline{T}_V - 1$$

Since t_1 and t_2 are given, we need only calculate $\overline{T}_V$.

Let $F_1(t)$ be the renewal time CDF for process 1 and $F_2(t)$ be the renewal time CDF for process 2. Then for the hypoexponential case these two CDF's are:

$$F_j(t) = \begin{cases} 0 & 0 \le t \le t_{j1} \\ 1-e^{-\epsilon_{j2}(t-t_{j1})} & t \ge t_{j1} \end{cases}$$

Where for $j = 1, 2$

$t_{j1} = t_j(1 - C_j) =$ deterministic time delay in figure A.1.a

$\epsilon_{j2} = \dfrac{1}{t_j C_j} =$ rate of exponential delay in figure A.1.a

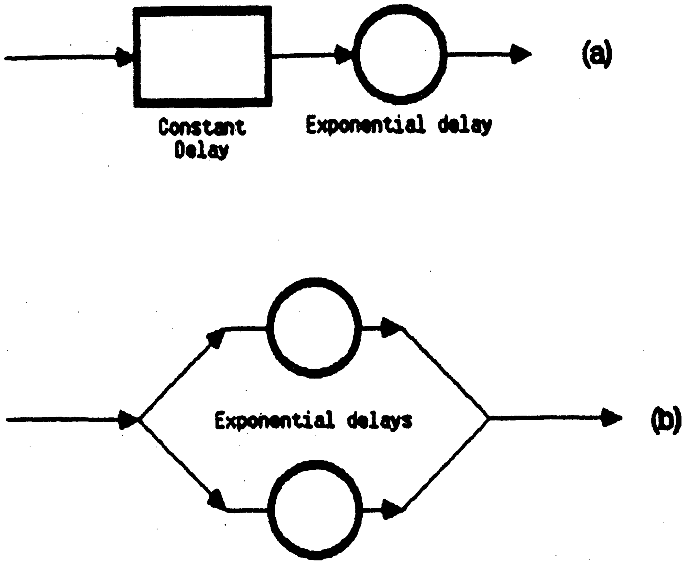


Figure A.1 Two moment representation function from [Kue79].

and if $C_j = 0 \Rightarrow \epsilon_{j2} = \infty$ or there is no exponential delay. Let $t_{j2} = 1/\epsilon_{j2}$.

$$T_V(t) = 1 - \frac{1}{t_1 t_2} (\int_t^{\infty}(1-F_1(u))du)(\int_t^{\infty}(1-F_2(u))du)$$

from [Kue79]. Then $\bar{T}_V = \int_0^{\infty}(1 - T_V(t))dt$. Rearrange the two processes so that $t_{11} \leq t_{21}$. Then

$$\bar{T}_V = \int_0^{t_{11}}(1-T_V(t))dt + \int_{t_{11}}^{t_{21}}(1-T_V(t))dt + \int_{t_{21}}^{\infty}(1-T_V(t))dt$$

Define

$$E_1 = \int_0^{t_{11}}(1-T_V(t))dt$$

$$E_2 = \int_{t_{11}}^{t_{21}}(1-T_V(t))dt$$

$$E_3 = \int_{t_{21}}^{\infty}(1-T_V(t))dt$$

Carry out the integrations as follows:

$t \leq t_{11} \leq t_{21} \Rightarrow$

$$T_V(t) = 1 - \frac{1}{t_1 t_2} \Big(\int_{t_{11}}^{\infty} e^{-t_{12}(u-t_{11})} du + t_{11} - t\Big)$$
$$\cdot\Big(\int_{t_{21}}^{\infty} e^{-t_{22}(u-t_{21})} du + t_{21} - t\Big)$$

$t_{11} \leq t \leq t_{21} \Rightarrow$

$$T_V(t) = 1 - \frac{1}{t_1 t_2} \Big(\int_{t}^{\infty} e^{-t_{12}(u-t_{11})} du\Big)$$
$$\cdot\Big(\int_{t_{21}}^{\infty} e^{-t_{22}(u-t_{21})} du + t_{21} - t\Big)$$

$t_{11} \leq t_{21} \leq t \Rightarrow$

$$T_V(t) = 1 - \frac{1}{t_1 t_2} \Big(\int_{t}^{\infty} e^{-t_{12}(u-t_{11})} du\Big)\Big(\int_{t}^{\infty} e^{-t_{22}(u-t_{21})} du\Big)$$

Then

$t \leq t_{11} \leq t_{21} \Rightarrow$

$$T_V(t) = 1 - \frac{1}{t_1 t_2} (t_1 - t)(t_2 - t)$$

$t_{11} \leq t \leq t_{21} \Rightarrow$

$$T_V(t) = 1 - \frac{1}{t_1 t_2} (t_{12} e^{-t_{12}(t-t_{11})})(t_2 - t)$$

$t_{11} \leq t_{21} \leq t \Rightarrow$

$$T_V(t) = 1 - \frac{1}{t_1 t_2} (t_{12} e^{-t_{12}(t-t_{11})})(t_{22} e^{-t_{22}(t-t_{21})})$$

$T_V(t)$ is continuous, $T_V(0) = 0$, $T_V(\infty) = 1$. Computing the three expectations E_1, E_2, E_3:

$$E_1 = \int_0^{t_{11}^{-}} (1 - T_V(t)) dt$$
$$= \frac{1}{t_1 t_2} \Big(\frac{t_{11}^3}{3} - \frac{(t_1 + t_2) t_{11}^2}{2} + t_1 t_2 t_{11}\Big)$$

$$E_2 = \int_{t_{11}}^{t_{21}^{-}} (1 - T_V(t)) dt$$
$$= \frac{t_{12}^2 e^{t_{11}/t_{12}}}{t_1 t_2} \Big[(t_{21} + t_{12} - t_2) e^{-t_{21}/t_{12}} - (t_{11} + t_{12} - t_2) e^{-t_{11}/t_{12}}\Big]$$

If $t_{12} = 0$, then assign $E_2 = 0$.

$$E_3 = \int_{t_{21}}^{\infty} (1 - T_V(t)) dt$$
$$= \frac{t_{22}^2 t_{12}^2 e^{\frac{t_{21} t_{12} + t_{11} t_{22}}{t_{12} t_{22}}}}{t_1 t_2 (t_{12} + t_{22}) e^{\frac{t_{21} t_{22} + t_{12} t_{21}}{t_{12} t_{22}}}}$$

If $t_{12} t_{22} = 0$, assign $E_3 = 0$. This completes the approximate calculations for the resulting coefficient of variation when two hypoexponential processes are superimposed.

BIBLIOGRAPHY

BIBLIOGRAPHY

[BaS76]
F. Baskett, and A. J. Smith, "Interference in Multiprocessor Computer Systems with Interleaved Memory," *CACM,* Vol. 19, No. 6, June 1976, pp. 327-334.

[Bha73]
D. P. Bhandarkar, "Analytic Models for Memory Interference in Multiprocessor Computer Systems," Ph.D. dissertation, Elec. Eng. Dept., Carnegie-Mellon Univ., Pittsburgh, PA, Rep. AD 773 843, Sept. 1973

[Bha75]
D. P. Bhandarkar, "Analysis of Memory Interference in Multiprocessors," *IEEE TC,* Vol. C-24, No. 9, Sept. 1975, pp. 897-908.

[Cin72]
E. Cinlar, "Superposition of Point Processes," *Stochastic Point Processes: Statistical Analysis, Theory, and Applications,* Peter A.W. Lewis, Ed. John Wiley and Sons, Inc. 1972, pp. 549-606.

[Cin75]
E. Cinlar, *Introduction to Stochastic Processes,* Prentice-Hall Inc., Englewood Cliffs, N.J., 1975.

[Fre82]
A. A. Fredericks, "A Class of Approximations for the Waiting Time Distribution in a GI/G/1 Queueing System," *Bell System Technical Journal,* Vol. 61, No. 3, March 1982, pp. 295-325

[GrH74]
D. Gross, and C. M. Harris, *Fundamentals of Queueing Theory,* John Wiley and Sons Inc., New York, 1974.

[GoG83]
A. Gottlieb, R. Grishman, C. P. Kruskal, K. P. McAuliffe, L. Rudolph, and M. Snir, "The NYU Ultracomputer--Designing an MIMD Shared Memory Parallel Computer," *IEEE TC,* Vol. C-32, No. 2, Feb. 1983, pp. 175-189.

[HeS82]
D. P. Heyman, and M. J. Sobel, *Stochastic Models in Operations Research, Vol. I: Stochastic Processes and Operating Characteristics,* McGraw-Hill, Inc., New York, 1982.

[Hoo77]
C. H. Hoogendorn, "A General Model for Memory Interference in Multiprocessors," *IEEE TC,* Vol. C-26, No. 10, Oct. 1977, pp. 998-1005.

[Kle75]
L. Kleinrock, *Queueing Systems Volume I: Theory,* John Wiley & Sons Inc., New York, 1975.

[Knu73]
D. E. Knuth, *The Art of Computer Programming, Vol. 3: Sorting and Searching,* Addison-Wesley Publishing Co., Inc., Reading, Mass., 1973.

[KrL76]
W. Kraemer, and M. Langenbach-Belz, "Approximate formulae for the delay in the queueing system GI/G/1," Congressbook, 8th Internat. Teletraffic Congress, Melbourne, 1976.

[Kue79]
P. J. Kuehn, "Approximate Analysis of General Queueing Networks by Decomposition," *IEEE Transaction on Communications,* Vol. COM-27, No. 1, Jan. 1979, pp. 113-126.

[MaG81]
M. A. Marsan, and M. Gerla, *Markov Models for Multiple Bus Multiprocessor Systems,* Report No. CSD 810304, Computer Science Department, UCLA, Feb. 1981.

[MaM82]
B. A. Makrucki, and T. N. Mudge, "A Stochastic Model of Parallel and Concurrent Program Execution on Multiprocessors," Computing Research Lab Report No. CRL-TR-3-82, Dept. of Electrical and Computer Engineering, University of Michigan, October 1982.

[McC73]
J. W. McCredie, "Analytic Models as Aids in Multiprocessor Design," *Proc. 7th Annual Princeton Conf. on Information and System Sciences,* March 1973, pp. 186-191.

[MuM82a]
T. N. Mudge and B. A. Makrucki, "Probabilistic Analysis of a Crossbar Switch," *Proc. 9th International Symposium on Computer Architecture,* IEEE, April 1982, pp. 311-320.

[MuM82b]
T. N. Mudge and B. A. Makrucki, "An Approximate Queueing Model For Packet Switched Multistage Interconnection Networks," *Proc. of the 3-rd Int. Conference on Distributed Computing Systems,* October 1982, (to appear).

[MuM82c]
T. N. Mudge and B. A. Makrucki, "Analysis of Multistage Networks with Unique Interconnection Paths," *Proceedings of the 14-th Southeastern Symposium on System* Theory, April 1982.

[Pat79]
J. H. Patel, "Processor-Memory Interconnections for Multiprocessors," *Proc. 6th*

UNIVERSITY OF MICHIGAN
3 9015 03465 7943

Annual Symp. on Computer Architecture, IEEE, April 1979, pp. 166-177.

[Ram65]
C. V. Ramamoorthy, "Discrete Markov Analysis of Computer Programs," *Proc. ACM 20th National Conference,* pp. 386-392.

[Rau79]
B. R. Rau, "Interleaved Memory Bandwidth in a Model of a Multiprocessor Computer System," *IEEE TC,* Vol. C-28, No. 9, Sept. 1979, pp. 678-681.

[Ros70]
S. M. Ross, *Applied Probability Models with Optimization Applications,* Holden-Day, Inc., San Francisco, 1970.

[SeD79]
A. S. Sethi, and N. Deo, "Interference in Multiprocessor Systems with Localized Memory Access Probabilities," *IEEE TC,* Vol. C-28, No. 2, Feb. 1979, pp. 157-163.

[SeM81]
K. C. Sevcik and I. Mitrani, "The Distribution of Queueing Network States at Input and Output Instants," *JACM,* Vol. 28, No. 2, April 1981, pp. 358-371.

[SiB82]
D. P. Siewiorek, C. G. Bell, and A. Newell, *Computer Structures: Principles and Examples,* McGraw-Hill, Inc., New York, 1982.

[SkA69]
C. E. Skinner, and J. R. Asher, "Effects of storage contention on system performance," *IBM Systems Journal,* No. 4, 1969, pp. 319-333.

[Smi74]
A. J. Smith, *Performance Analysis of Computer System Components,* Ph.D. Thesis, STAN-CS-74-451, Computer Sci. Dept., Stanford Univ., August 1974.

[Smi82]
A. J. Smith, "Cache Memories," *ACM Computing Surveys,* Vol. 14, No. 3, Sept. 1982, pp. 473-530.

[Str70]
W. D. Strecker, *Analysis of the Instruction Execution Rate in Certain Computer Structures,* Ph.D. dissertation, Carnegie-Mellon University, Pittsburgh, 1970.

[Whi83]
"Toward an Approximation Theory for Point Processes and Networks Queues," *Newsletter of the ORSA/TIMS Applied Probability Group,* Fall 1983, pp. 2-4.