Performance Evaluation of Distributed Systems with Unbalanced Flows: An Analysis of the INFOPLEX Data Storage Hierarchy

by

Yng-Yuh Richard Wang

B.A. National Taiwan University
(1975)

M.B.A. University of Wisconsin, Madison
(1979)

Submitted to the Sloan School of Management

in Partial Fulfillment of the

Requirement of the Degree of Doctor OF PHILOSOPHY

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

July 1984

Signature of Author: ______________________________
Sloan School of Management, July 1984

Certified by: ______________________________
Thesis Supervisor

Accepted by: ______________________________
Chairman, Ph.D. Committee

Performance Evaluation of Distributed Systems with Unbalanced Flows: An Analysis of the INFOPLEX Data Storage Hierarchy

by

Yng-Yuh Richard Wang

Submitted to the Sloan School of Management on July, 1984 in partial fulfillment of the requirements for the degree of doctor of philosophy at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

ABSTRACT

A software engineering methodology to evaluate systems performance early in the design process is presented. Specifically, a technique is presented to compute performance measures for distributed systems with unbalanced flows due to asynchronously spawned parallel tasks -- a common phenomenon in modern information systems which results in a primary effect on performance. With this technique, a cost effective tool can be developed to analyze an architectural design and produce measures such as throughput, utilization, and response time so that potential performance problems can be identified and erroneous design decisions reduced. An algorithm based on Buzen's convolution algorithm has been developed to test the necessary and sufficient conditions for system stability as well as to compute the closed system throughput. An average of less than four iterations has been reported for the efficient algorithm. A comparative study of the INFOPLEX data storage hierarchy using TAD, a cost effective tool based on this iterative algorithm, versus detailed simulations has been conducted and highly consistent results have been observed.

Thesis Supervisor: Dr. Stuart E. Madnick
Title: Associate Professor of Sloan School of Management

TABLE OF CONTENTS

Acknowledgement

To a great advisor and friend, Prof. Stuart Madnick, for his support, inspiration, and supervision.

To the members of the thesis committee, Prof. William Frank, Prof. Ugo Gagliardi, and Prof. Anthony Wong, for their encouragement, criticism, and enthusiasm.

To the Sloan School of Management, the Center for Information Systems Research, and the International Business Machines Corporation for their scholarships.

To my family and friends for their care and patience.

CHAPTER I

INTRODUCTION AND PLAN OF THESIS

I.1 GOAL OF THESIS

The goal of system development is to produce systems that satisfy their specifications when completed while minimizing costs and time required. The main key to minimizing costs and time is to determine whether the system will meet its functional and performance requirements as early as possible in the development process. This will avoid wasted work toward an unsatisfactory implementation and the subsequent rework. To this end, a cost effective tool to evaluate system performance is essential (see reference 32).

The primary goal of this thesis is to provide a software engineering methodology for evaluating the performance of distributed systems with unbalanced flows due to asynchronously spawned parallel tasks early in the design process. Specifically, it aims to provide insight into and shed additional light on the performance problems inherent in the design and analysis of the INFOPLEX data storage hierarchy. (INFOPLEX is a database computer research project being conducted at the Center for Information Systems Research, Massachusetts Institute of Technology (M.I.T.); the theory of hierarchical decomposition is applied in this research to structure hundreds of microprocessors together to realize a low cost data storage hierarchy with very large capacity and minimum access time.)

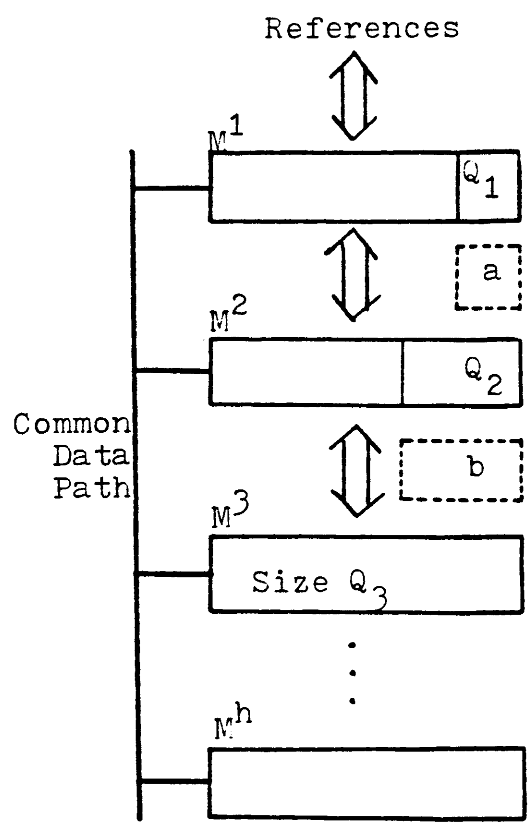
I.2 SIGNIFICANCE OF PROBLEM

I.2.1 Cost effectiveness

Unbalanced flows due to Asynchronously spawned Parallel tasks (UAP) is a common phenomenon in modern information systems utilizing distributed processing or local area networking. As a result, it has a primary effect on the system's performance. However, this kind of phenomenon can not be analyzed by classical product form queueing network models. In the remainder of this thesis, the acronym UAP will refer to unbalanced flows due to asynchronously spawned parallel tasks which are assumed to run independently of each other except for resource contention.

To make the problem more concrete and realistic, the author illustrates the broadcast phenomenon with the INFOPLEX data storage hierarchy model (1, 46, 47, 55, 56, 93, 94):

A data storage hierarchy consists of h levels of storage devices, M^1, M^2, ..., M^h. The page size of M^i is Q_i and the size of M^i is m_i pages each of size Q_i. Q_i is always an integral multiple of Q_{i-1}, for i = 2, 3 ..., h. The unit of information transfer between M^i and M^{i+1} is a page, of size Q_i. Figure I.1 illustrates this model of the data storage hierarchy.



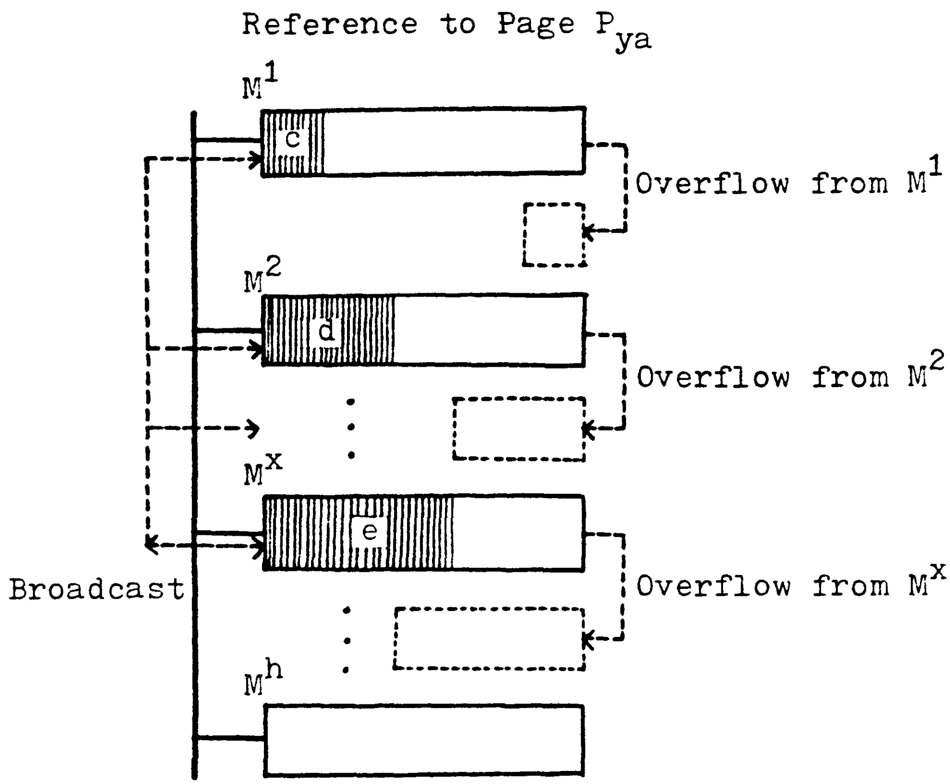
a: Unit of Data Transfer between M^1 and M^2
b: Unit of Data Transfer between M^2 and M^3

Figure I.1 Model of a Data Storage Hierarchy

There are two basic operations in the data storage hierarchy: the READ-THROUGH operation and the STORE-BEHIND operation. The author will use the READ-THROUGH operation to illustrate broadcast and refer the reader to Lam (46) for STORE-BEHIND to illustrate acknowledgement. In a READ-THROUGH operation, the highest storage level that contains the addressed information broadcasts the information to all upper storage levels, each of which simultaneously extracts the page (of the appropriate size) that contains the information from the broadcast. If the addressed information is found in the highest storage level, the READ-THROUGH reduces to a simple reference to the addressed information in that level. Figure I.2 illustrates the READ-THROUGH operation. A corresponding queueing network model of the broadcast is shown in Figure I.3. Note that the routing probabilities out of queue M^x equals (X-1) which is greater than one.

Since READ-THROUGH and STORE-BEHIND are the two fundamental operations in the INFOPLEX data storage hierarchy, broadcast and acknowledgement produce a significant portion of load to devices. It is critical to incorporate this unbalanced flow into the performance model.

Simulation models have been used to evaluate performance of this kind of system (46). A major disadvantage of simulation models is the prohibitive cost incurred in obtaining performance measures for different design alternatives.



c: Page P_{ya}
d, e: Page containing P_{ya}

<u>Figure I.2</u> <u>The READ-THROUGH Operation</u>

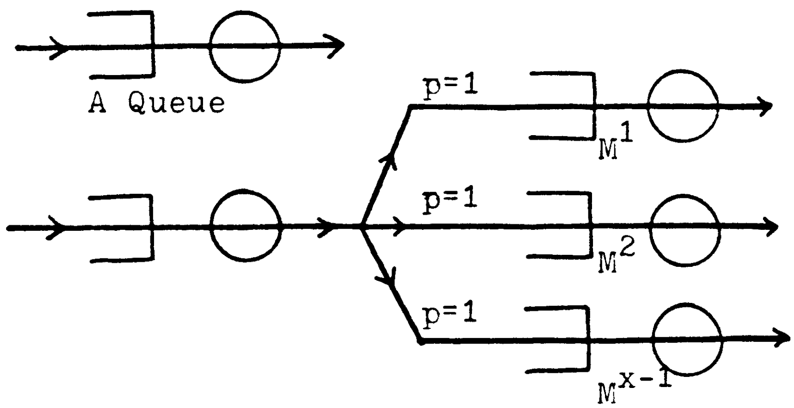


Figure I.3 READ-THROUGH Broadcast Queueing Diagram

Figure I.4 depicts the difference in terms of CPU time and dollar cost between the simulation model and the analytic model (based on the technique developed in this thesis) that the author has conducted for the INFOPLEX P5L4 (5 processors, 4 levels) model. Clearly it pays off to employ the analytic model instead of the simulation model in exploring different design alternatives if consistent results can be obtained from the analytic model.

RUN	SIMULATION PERIOD	SIMULATION CPU-TIME	SIMULATION COST$	ANALYTIC CPU-TIME	ANALYTIC COST$
1	10 ms	434	97.33	12	0.05
2	3 ms	270	61.70	12	0.05
3	2 ms	349	78.22	12	0.05
4	2 ms	308	70.32	12	0.05
5	1 ms	205	47.77	12	0.05
6	1 ms	351	79.02	12	0.05
7	.5 ms	453	101.06	12	0.05
8	.3 ms	290	65.55	12	0.05
9	.05 ms	47	13.09	12	0.05
10	.05 ms	38	10.54	12	0.05

```
COST$
120.+
    -
    -                                              S
    -                                            S
    -
 80.+                                    SS
    -                                S
    -                            S S
    -
    -                      S
 40.+
    -
    -
    -      S
    -     S
  0.+     AA               A     A A A   AA      A A
      +---------+---------+---------+---------+---------+CPU-TIME
      0.      100.      200.      300.      400.      500.
```

1 Simulation CPU-TIME is in CPU seconds on an IBM 370/168.
2 Analytic CPU-TIME is 12 CPU seconds per run on a PRIME/850.
3 An IBM 370/168 is about 5 times faster than a PRIME/850.
4 "Cost$" is in dollars for the overall charge per run.
5 "ms" in the table means milli-seconds.
6 To attain steady-state, simulation periods of 10 ms, or more, are usually needed.

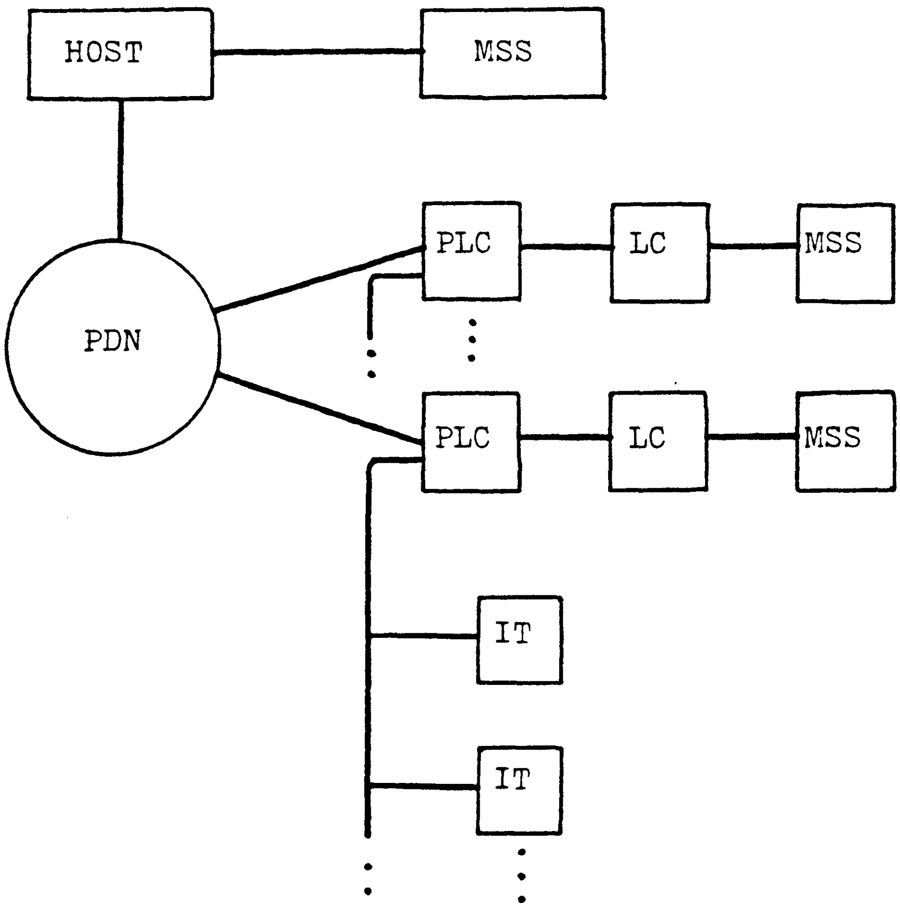
Figure I.4:
A Comparison of the costs: Analytic(A) vs Simulation(S).

I.2.2 Impact upon System Development

A more fundamental issue, in addition to cost-effectiveness, is the significance of performance evaluation to system development. This issue is addressed with a case (32) which reviews a large retail front and back office banking system. In the development of this system, system performance analysis was not conducted. Consequently, after the prototype of the system was implemented, serious performance problems arose.

This system is a simple two-level tree structured network with a root of a mainframe host facility and a large mass storage subsystem. There are 1300 first level nodes of local computers (minicomputers) with local mass storage; 5600 second level nodes of intelligent terminals (microcomputers) without local mass storage.

The connection between the host and the local computers is established through a packet switching public data network; the connection between the local computers and the intelligent terminals is through very high speed local lines (Ethernet like protocol) and a programmable line controller (PLC). The PLC handles the local computer connection to both the packet data network and the local line; in other words, all local computer traffic goes through the PLC, as shown in Figure I.5.



MSS: Mass Storage Subsystem
PDN: Packet Data Network
PLC: Programmable Line Controller
LC : Local Computer
IT :Intelligent Terminal

Figure I.5 System Configuration of Case I.2.2

All customer information is centralized at the host site; information at the local computer site is limited to access control, forms, and application programs; in addition, no mass storage is allowed at the intelligent terminal. The motivation for the design decision was twofold:

A. Keep the host subsystem common to the old and the new system; the old system had no local computers and used dumb terminals. By centralizing all customer information at the host site, compatibility is preserved.

B. Keep the cost of terminals as low as possible. By eliminating mass storage at the intelligent terminal level, it was believed that costs could be reduced.

This led to an inordinate amount of traffic up and down the tree. In order to keep the local computer cost down, it was further decided to handle all local computer traffic through a single PLC (as mentioned before). The consequence of this design is a major bottleneck at the PLC.

The lesson from the case is that all decisions should be made as a rational and quantitative design activity instead of by management fiat. After a posteriori quantitative performance analysis in the review, it was recommended that some mass storage be allocated at the intelligent terminal to relieve the traffic generated by form and record requests from the intelligent terminal to the local computer.

It should be pointed out that the system designers were not unintelligent. Their mistake was the result of a lack of guidance, methodologies, and appropriate tools to support their design and decision activities. Had a cost-effective performance analysis tool been employed during the system development process to serve as the alter ego for functional analysis, the serious performance problem would not have occurred (32).

In sum, the significance of the problem lies in the necessity of performance analysis to the success of system development and the importance of cost-effective tools to the performance analysis of different design alternatives.

I.3 ACCOMPLISHMENTS OF RESEARCH

The specific accomplishments of this research, which will be elaborated upon later, are:

* Model and analyze distributed systems with unbalanced flows.
* Investigate the existence of a product form solution for distributed systems with unbalanced flows.
* Develop an analytic formulation for open systems.

* Develop an efficient iterative algorithm to test the necessary and sufficient condition for closed system stability as well as to compute the closed system throughput.
* Model and analyze distributed systems with unbalanced flows and priorities.
* Implement a software package to evaluate performance of the INFOPLEX data storage hierarchy.
* Validate the theory using the INFOPLEX data storage hierarchy models.
* Explore different design alternatives for the INFOPLEX data storage hierarchy based on the results of technology analyses.

I.4 STRUCTURE OF THESIS

This thesis is divided into eight chapters. The content of the chapters, and thus the structure of the thesis, are delineated below.

Chapter II: Performance Evaluation of Computer Systems Using Analytic Queueing Networks

This chapter presents a perspective on state-of-the-art performance evaluation using analytic queueing network models. It reviews the literature and the background theory necessary for the remainder of the thesis. It is targeted primarily at readers knowledgeable in the design and analysis of computer systems but who are not specialized in queueing theory. Those familiar with queueing theory may skip this chapter.

Chapter III: Existence of The Product Form Solution for Systems with Unbalanced Flows

The product form solution for the equilibrium state probabilities of queueing network models was first presented by J.R. Jackson in 1957 (42). This result has been extended by many researchers since then (5, 7, 8, 18, 20, 21, 22, 35, 61, 64, 67, 69, 70, 85, 86, 89) and summarized by Chandy in 1980 (22). By a flow conservation argument, it has been shown that the product form solution exists for a certain class of queueing network models (5). This result is rather surprising as Burke points out since the arrival process to a service facility is not Poisson in general (7).

A crucial question to ask is whether the product form solution also exists for systems with unbalanced flows, assuming a certain physical characteristic holds which allows flows not to be conserved at the flow unbalanced points. The answer to this question is important from the theoretic point of view. On the one hand, if it is proven that the product form solution does exist, then the breakthrough will extend the product form theory to the flow unbalanced networks; on the other hand, if it is shown that the product form solution does not exist in general, then one has to use other techniques. An analogy to this is that if it is shown that a problem is NP-complete, then one can employ heuristic algorithms to solve the problem. This question is addressed with a counter example to show that product form solution does not exist in the example with our assumptions.

Chapter IV: Modeling and Analysis of Distributed Systems with Unbalanced Flows

This chapter presents a description of the model and an analytic formulation of distributed systems with unbalanced flows. A mathematical treatment is given to address the following topics.

ANALYTIC FORMULATION

An analytic technique for systems with unbalanced flows is presented to obtain performance measures. With this technique, a cost-effective tool can be developed to analyze an architectural design and to produce measures such as throughput, utilization, and response times so that potential performance problems can be identified to reduce erroneous design decisions.

NECESSARY AND SUFFICIENT CONDITIONS FOR CLOSED SYSTEM STABILITY

This condition is investigated and identified. It is employed to determine whether a system will be stable with a given set of parameters. If it is insured that the stability condition exists, then an efficient iterative algorithm is applied to locate the equilibrium system throughput. Moreover, it provides insight into the behavior and structure of the system and helps system designers to locate good design alternatives.

EFFICIENT ITERATIVE ALGORITHM FOR CLOSED SYSTEMS

The algorithm is used to locate the equilibrium system throughput as well as the corresponding normalization constant. Once these two values are known, other performance measures follow (71).

PRIORITY TREATMENT OF DISTRIBUTED SYSTEMS WITH UNBALANCED FLOWS

A solution to treat the unbalanced flows with a different priority from the main flow is presented in this section. It provides further insight into the behavior of the INFOPLEX data storage hierarchy where the STORE-BEHIND operation consumes a great deal of resources and may be handled with a lower priority.

Chapter V: Efficiency of Iterative Algorithms and Implementation of TAD

The efficiency of iterative algorithms are investigated in this chapter Moreover, a software package called TAD (Technique for Architectural Design) for the INFOPLEX data storage hierarchy is presented to demonstrate the practicality of this research.

ITERATIVE ALGORITHMS

The iterative algorithm is based on Buzen's convolution algorithm which evaluates the normalization constant of the product form solution. It has been observed, during more than 2400 simulations, that the procedure takes an average of 4 iterations to produce a relative error of less than 0.001 given an initial estimate. The converging speed of the iterative algorithm is shown to be $\log_4$ based and the computational efficiency of each iteration is the order of M*N (o(MN)) where M is the number of service facilities and N is the number of customers in the system.

TAD

Salient features which are unique to TAD include: a) the efficient procedure mentioned above to test the necessary and sufficient condition for closed system stability and to iteratively compute the closed system throughput; b) an efficient procedure to eliminate the routing definitions and to calculate the visit ratios of a data storage hierarchy; and c) a user friendly interface with menu-driven inputs and graphic outputs to adapt to the INFOPLEX data storage hierarchy.

In addition to ease of use, it has been observed that use of TAD costs five cents per design alternative; on the other hand, it would cost hundreds of dollars to obtain the desired information using simulation. To be specific, one can use TAD to explore 2000 design alternatives at a cost of $100. Whereas, it may not be possible to attain steady-state results of a single design alternative using simulation for $100.

Chapter VI: Validation Study Using the INFOPLEX Data Storage Hierarchy Models

The validation of the analytical formulation is presented in this chapter through RESQ and GPSS simulation models (48, 79) using the INFOPLEX P1L3 and P5L4 models. It has been observed that the analytic results are highly consistent with the simulations. A closer examination of the data shows that the results were accurate with a relative error of less than 2%.

Chapter VII: Technology Analysis and Design Alternative Exploration

Processor and storage technologies for 1984 and 1988 are investigated and projected in this chapter. These raw data are used as input to TAD to explore different design alternatives of the INFOPLEX data storage hierarchy. Problems such as the ratio of read vs. write operation to the performance of the data storage hierarchy, and the impact of locality to the performance of the data storage hierarchy are investigated.

Chapter VIII: Summary and Conclusions

In addition to a general summary of the significant aspects of the thesis, this chapter outlines important areas for future research.

CHAPTER II

Performance Evaluation of Computer Systems Using Analytic Queueing Network Models

II.1 MOTIVATION FOR USING ANALYTIC PRODUCT FORM QUEUEING NETWORK MODELS

An IBM PC user who runs a MS/DOS 1.0 would enjoy full access to all system resources such as CPU, memory, and disks. A major disadvantage of the system, though, is the inefficiency of utilization of the system resource. For instance, the IBM PC user would not experience the excitement of observing the printer printing, the disk drive lights flashing, and the presentation graphics program displaying animated cartoons at the same time.

This was what happened prior to the advent of multiprogramming systems. In the late 50's, computers became commercially available and multiprogramming was introduced to improve the efficiency of utilization of system resources by allowing multi-users to gain access to the system. However, this gave rise to contention for resources among competing users and led to queueing delays. Since the queueing delays may cause significant deterioration in the system performance, researchers began to use queueing models to study the queueing effects on the performance of computer systems (3). In particular, queueing network models, which have product form solutions, received considerable attention because they made feasible the study of networks with many service facilities and/or large populations.

Some issues of Computing Surveys (28, 37) and Computer (3) have focused on the solution of product form queueing network models and the representation of computer and communication network systems as queueing networks (95).

A product form queueing network is one that has a solution in the following form:

$$P(S_1,\ldots,S_M) = P_1(S_1) \ldots P_M(S_M)/G(N)$$

where $P(S_1, \ldots S_m)$ is the steady-state probability of a network state in a network with M service facilities, $P_m(S_m)$, $m = 1, \ldots, M$ is the probability that the m_{th} service facility is in state S_m in isolation. N is the number of customers in the network, and G(N) is a normalization constant. For an open system, N can be any number; for a closed system, N is a fixed number of customers in the system. The normalization constant G(N) is equal to the sum of $P_1(S_1) * \ldots * P_m(S_m)$ over all feasible network states.

If a queueing network model does not have a product form solution, then we usually must use fairly general numerical techniques, such as solution of Markov balance equations, for its solution. In this case we shall find the exact solution of the network intractable unless it has few service facilities and/or customers (49).

II.2 LITERATURE REVIEW

The product form solution for the equilibrium state probabilities of queueing network models was first introduced by J.R. Jackson in 1957. In 1963, Jackson extended his analysis to open and closed systems with local load-dependent service rates at all service facilities (42). Gordon and Newell restructured the result for the closed system (35). In 1971, Buzen presented a fast computational algorithm, known as convolution algorithm, to compute the normalization constant for closed systems (14). In 1975, Baskett, Chandy, Muntz, and Palacious extended the results to include different queueing disciplines, multiple classes of jobs, and non-exponential service distributions (5); their results are known as the BCMP theorem. Chandy provided a summary of the product form theory in 1982 (22). These results are based on traditional stochastic analysis of queueing networks. An alternative framework, Operational Analysis for studying queueing systems, was introduced by Buzen in 1976 and elaborated subsequently (8, 9, 10, 11, 12, 16). This approach is based on assumptions about the deterministic behavior, over a finite time interval, of the system being modeled. Using the operational approach, one can obtain the same product form solution for closed networks but with nonprobabilistic assumptions about the network. Instead of obtaining the steady-state probability of a network state, one obtains the fraction of the time interval that the network is in a state (28). Operational Analysis provides us with many of the informal, intuitive arguments about the behavior of queueing networks (indeed the technique presented in this thesis was first perceived in the context of Operational Analysis); on the other hand, the traditional stochastic analysis provides a solid basis for the theoretical development of new results. In this thesis, the stochastic approach is adopted.

The first successful application of a queueing network model to a computer system was made in 1965 when Sherr used the classical machine repairman queueing model to analyze the MIT time sharing system, CTSS. In 1971, Buzen introduced the central server model. Working independently, Moore showed that queueing network models could predict the response times in the Michigan Terminal System (MTS) to within 10% error (28). Since then, the use of analytical performance models instead of simulation models has become much more popular. Graham (37) summarized some of the basic reasons for this as follows:

1. These models capture the most important features of actual systems. Experience shows that performance measures are much more sensitive to parameters such as mean service time per customer at a service facility than to many of the details of policies and mechanisms throughout the operating system (which are difficult to represent concisely).

2. The assumptions of the analysis are realistic. General service time distribution can be handled at many

service facilities; load dependent facilities can be modeled; and multiple classes of customers can be accommodated.

3. The algorithms that solve the equations of the model are available as highly efficient queueing network evaluation packages.

Another very important reason for the increasing popularity of these models is simple: they work.

In order to obtain consistent results, the primary effects on performance should be captured in the analytic model. UAP has been found to be one of the primary effects on performance (93, 94). Unfortunately, networks with UAP did not have an analytically tractable solution because the input flow and the output flow are not balanced at the places where parallel tasks are spawned, a violation of the principle of job flow balance (28) (The principle of job flow balance states that the number of customers that flow into a service facility equals the number of customers that flow out of the facility when the system is in the steady-state.)

A simplified INFOPLEX P1L2 (one processor, 2 levels) data storage hierarchy model is given below to illustrate the UAP phenomenon.

Example:

Consider the routing diagram (Figure II.1) of a simplified P1L2 data storage hierarchy which processes the read and write operations. Suppose 80% of the customers request the read operation (class RP1) and 20% request the write operation (WP1); and the read operation has 100% locality, i.e. data are always found at D1. The read operation is serviced by the level one processor P1 first, then retrieved from D1 and returned to the reference source (SINKM). The write operation is acknowledged immediately by P1 to the reference source (SINKM); in parallel, the data are updated at D1, stored-behind to the level 2 device D2, then the asynchronously spawned task terminates (SINKU).

Note that class WP1 leaves facility P1 with a routing probability one to SINKM and a routing probability one to WD1 as indicated by the dash line, i.e. the out-flow is twice as much as the in-flow, violating the principle of flow balance.

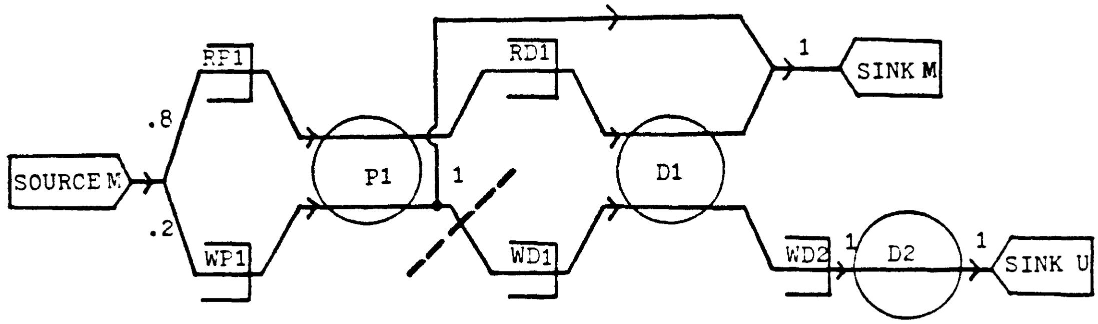


Figure II.1 Routing Diagram for P1L2 Model

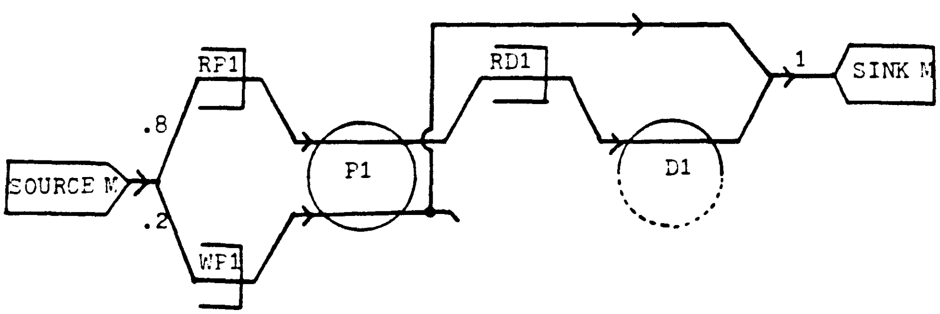


Figure II.2 Main Chain

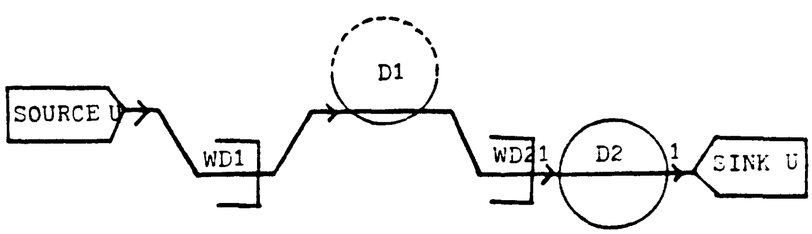


Figure II.3 UAP Chain

Several studies have attempted to generalize queueing network models to include parallel processing. Browne, Chandy, Horgarth, and Lee (6) investigated the effect on throughput of multiprocessing in a multiprogramming environment using the central server model approach. Sauer and Chandy (71) studied the impact of distributions and disciplines on multiple processor systems. Towsley, Chandy, and Browne (87) developed approximate queueing models for internal parallel processing by individual programs in a multiprogrammed system based on the central model approach and the "Norton theorem." Price (63) analyzed models of multiple I/O buffering schemes. Others (59, 62) modeled a number of CPU:IO overlap cases. These studies, although valuable, do not fit systems which 1) have a generalized topology, and 2) have the UAP phenomenon.

Modeling the UAP phenomenon for generalized queueing network systems is a relatively new topic, first reported, to the author's knowledge, by Heidelberger and Trivedi in 1982 (39). In that work, An approximate solution method is developed and results of the approximation are compared to those of simulations. Mean value analysis approximation techniques are proposed for local area distributed computer systems with UAP by Goldberg, Popek, and Lavenberg (34).

It is perhaps interesting to note at this point that, quite independently from the above research, the author developed what is known as "Flow unbalanced general queueing network analysis"

(93, 94) starting in 1981. The technique used to model UAP is very similar but a different algorithm has been used to test the necessary and sufficient condition as well as to compute the closed network throughput. Moreover, the results for open networks with UAP, such as response time, have been analyzed in the INFOPLEX research. A syntactic definition has also been given to decompose a model uniquely.

A terminal-oriented system and a batch-oriented multiprogramming system were modeled by Heidelberger (39), and local area distributed systems were modeled by Goldberg and others (34) while a hierarchically decomposed architecture is modeled in the INFOPLEX research (93, 94). The consistency reported from modeling these different architectures provides further validation of the modeling technique. The background theories which are essential for the remainder of the thesis are reviewed below.

II.3 BACKGROUND THEORY

Notations used in this section and the remainder of the thesis are listed below:

A) subscripts:

i denotes an individual service facility.
o denotes the overall network.
(M) denotes the main chain.
(U) denotes the UAP chain.
$()^i$ denotes the ith iteration.

B) notations:

B bottleneck facility (therefore chain) throughput.

C	total number of classes in the network.
CMD	continuous and monotonically decreasing
D	V*S; the product of visit ratio and mean service time.
FCFS	first come first serve.
f	$X_o(M)=f(X_o(U))$; the main chain throughput as a nonlinear function of the UAP chain throughput.
IS	infinite server.
LCFSPR	last come first serve preemptive resumable.
M	number of service facilities in the network.
N	mean number of customers (mean queue length including the one in service).
n	number of customers.
PS	processor sharing.
p.f.s.	product form solution.
p.g.f.	probability generating function.
R	mean response time.
S	mean service time.
U	utilization.
UAP	unbalanced flows due to asynchronously spawned parallel tasks.
V	visit ratio.
X	throughput.
λ	arrival rate.
μ	service rate.
ρ	traffic intensity.

Example: $S_i(M)$ means the mean service time of facility i for the main chain; $V_i(M)$ means the visit ratio to facility i due to the main chain; and $D_i(M) = S_i(M) * V_i(M)$ is the product of visit ratio and mean service time of facility i for the main chain.

The analytic approach of performance evaluation of distributed systems requires a great deal of background knowledge in queueing theory. To present the thesis concisely, only the most relevant results are presented in this section. A comprehensive bibliographic list is appended for those interested in this area.

II.3.1 Little's Formula

Let N be the average over all time of the number of customers in a system, λ be the average arrival rate at the system, and R be the average over all arrivals at the system of the system response time, then $N = \lambda * R$. This formula states that the average number of customers in the system is equal to the product of the arrival rate and the average system response time.

II.3.2 Product Form Queueing Networks (PFQN)

For the following queueing disciplines, a product form solution exists for a queueing network: first come first serve (FCFS), processor sharing (PS), infinite server (IS), and last come first serve preemptive resumable (LCFSPR). If a server has a PS, IS, or LCFSPR discipline, then different service time distributions are allowed for different classes at a service facility. In this case, the service time distributions affect the performance measures we shall consider only through the mean service time. If a service facility has a FCFS discipline, then all classes at the facility must have the same exponential service time distribution (5).

II.3.3 Single Chain Queueing Networks (SCQN)

A single chain queueing network is one with only one customer type. However, service facilities may have several classes which allow customers to have different sets of routing probabilities for different visits to a service facility. Note that although there are several classes and several routing probabilities, the only parameters in the product form solutions, when aggregated to the service facility level, are visit ratios, mean service times, and number of customers in the closed queueing network case (49).

II.3.4 Open Product Form Single Chain Queueing Networks (OPFSCQN)

An OPFSCQN is one with M service facilities and C classes and a single chain that has a product form solution. In addition, there are sources for exogenous arriving customers and sinks for departing customers. It is assumed that customers from exogenous sources form a Poisson process with a constant arrival rate λ.

A remarkable theorem by Jackson states that for OPFSCQN with a constant arrival rate, the network is separable (42), i.e. one can compute a service facility's performance measures as follows (28, 49, 71): Suppose the probability that an arrival customer enters class c is $P_{O,c}$ then it must be true that

$$\sum_{i=1}^{C} P_{O,j} = 1$$

$$V_j = p_{o,j} + \sum_{j=1}^{C} V_i * p_{i,j} \qquad j = 1,\ldots,C$$

Suppose the system is in the steady-state, then the system arrival rate is equal to departure rate. Let X_0 denote system throughput, it follows that $X_0 = \lambda$. Let X_i be the throughput of facility i, it follows that $X_i = X_0 * V_i \quad i = 1, \ldots, M$. Let $U_i = X_i * S_i$ where U_i is the utilization of service facility i and S_i is the mean service time of facility i. It is easy to see that an open queueing network is stable iff $U_i < 1$ for all service facilities in the network. The IS discipline is excluded from our discussion to avoid unnecessary digression. The mean queue length (including the one in service) is $N_i = U_i / (1-U_i)$.

By Little's formula, the mean response time of service facility i is $R_i = N_i / X_i$. It follows that system response time $R = R_1 + \ldots + R_M$. The mean number of customers in the network $N = R / X_0$. Note that different formulae should be used for the IS discipline. Thus, for OPFSCQN, one can obtain system as well as facility throughput, response time, and mean queue length.

II.3.5 Open Product Form Multiple Chain Queueing Networks (OPFMCQN)

OPFSCQN have a single source and a single sink and all classes are reachable from the source and the sink is reachable from all classes. It is not necessary, however, that all classes be reachable from one another. If there are H sources and the classes are partitioned into H disjoint subsets such that for $h = 1, \ldots, H$, all classes in subset h are reachable from source h and not reachable from any other sources or any other classes in any other subsets, then there are H open routing chains (49). It can be shown (49, 71, 64) that if we have H chains, each with a Poisson source with a constant rate λ_h , $h = 1, \ldots, H$, then we can treat the H open chains as a single aggregate chain if we give that aggreate chain an arrival rate $\lambda = \lambda_1 + \ldots + \lambda_H$, and where class c belongs to chain h in the original network, make the replacement $P_{o,c} = (\lambda_h/\lambda)*P_{o,c}$, $c = 1, \ldots, C$.

II.3.6 Closed Product Form Single Chain Queueing Networks (CPFSCQN)

A closed product form single chain queueing network is one with M service facilities, C classes, and a fixed number of homogenous customers that has a product form solution. Several algorithms are available for CPFSCQN; the convolution algorithm (14) remains the dominant algorithm for general purpose use (49).

The equilibrium distribution of customers in CPFSCQN, aggregated at the service facility level, is given by:

$$P(n_1, \ldots, n_m) = (1/G(N)) * \prod_{i=1}^{M} (D_i)^{n_i}$$

where $D_i = V_i \cdot S_i$, and n_i is the number of customers of facility i. It can be shown (9) that

$$P(n_i = k) = (D_i)^k (G(N-k) - D_i * G(N-k-1))/G(N)$$

where G(n) is defined as zero for n<0.

The mean queue length of facility i, N_i, is given by

$$N_i = \sum_{k=1}^{N} (D_i)^k * G(N-k) / G(N)$$

The system throughput, X_o, is given by $X_o = G(N-1)/G(N)$. Therefore, once the values of G(1), ..., G(N) are given, a number of useful performance measures can be computed.

II.3.7 Convolution Algorithm

The expression for G(N) in the equilibrium distribution equation involves the summation of C(M+N-1,N) terms, each of which is a product of M factors which are themselves powers of the basic quantities. However, the celebrated convolution algorithm computes the entire set of values G(1), ..., G(N) using a total of N*M multiplications and N*M additions. The implementation of the algorithm is extremely simple:

```
/* Initialization */
```

```
G(0) = 1
for n = 1 to N
G(n) = 0
/* convolution */
for m = 1 to M
for n = 1 to N
G(n) = G(n) + D(m)*G(n-1)
/* end convolution */
```

II.3.8 Product Form Mixed Queueing Networks (PFMQN)

Let's restrict a product form mixed queueing network to be one with only one closed chain and one open chain. Let "(C)" denote the closed chain, and "(O)" denote the open chain. The traffic intensities of facility i due to the open chain and the closed chain are defined as

$$\rho_i(O) = X_o(O) * V_i(O) * S_i(O)$$

$$\rho_i(C) = X_o(C) * V_i(C) * S_i(C)$$

The p.g.f. method has been used by Reiser and Kobayashi (64) to provide important theoretical results for PFMQN. It was found, with the p.g.f. method, that

1) The stability of PFMQN is unaffected by the presence of closed chains;

2) The open and the closed chains do not interact at an IS service facility;

3) For FCFS, PS, and LCFSPR disciplines, the effect of the open chain on the closed chain is to increase the traffic intensity by $(1-\rho)^{-1}$; and

4) The closed chain throughput is evaluated through a nonlinear function of the open chain throughput.

CHAPTER III

Existence of the Product Form Solution for Systems with Unbalanced Flows

III.1 MOTIVATION AND SIGNIFICANCE

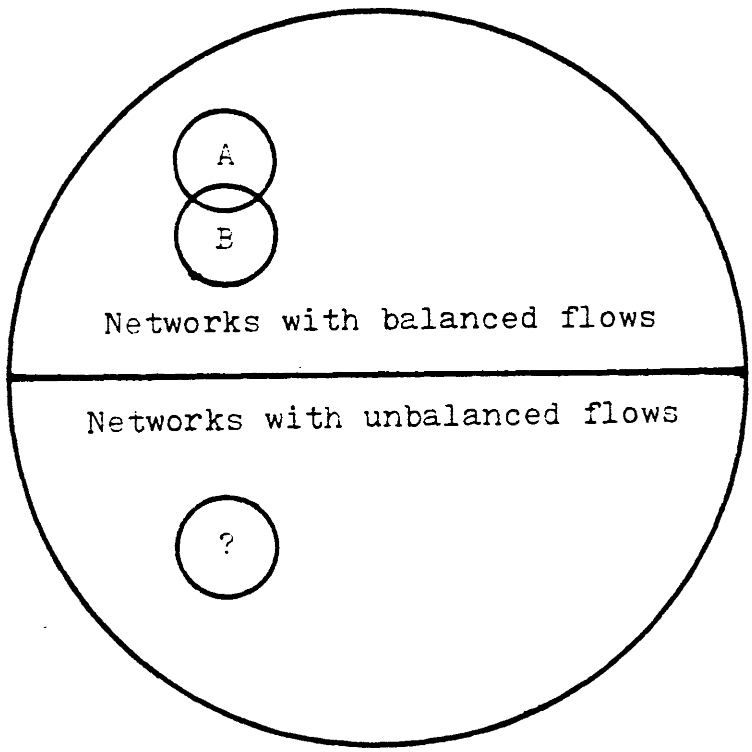
As mentioned in Chapter I.4, a crucial question is whether the product form solution also exists for systems with unbalanced flows assuming a certain physical characteristic holds which allows flows not to be conserved at the flow unbalanced points. It is logical to ask this question considering the derivation of the product form solution. As Burke pointed out, for a Jackson type queueing network, the combined input to a service facility, new arrivals and returning customers, is apparently not Poisson in general; nonetheless, Jackson found, by the flow conservation argument, that the steady-state joint probability distribution of the network with feedback is the product of individual service facility probability distributions -- a result which is astonishing in light of Burke's results (7).

A similar situation has been observed in systems with unbalanced flows by Madnick (58): while the combined input to a service facility in a Jackson type queueing network with balanced flows is not Poisson in general, the output process at the flow unbalanced points in a network with unbalanced flows is also not Poisson in general. It might be possible to apply some kind of techniques such as the one employed by Jackson to show that the product form solution also exists for systems with unbalanced flows given a set of reasonable assumptions.

This question is important from the theoretical point of view as was stated in Chapter I.4 and is recapitulated here: on the one hand, if it can be shown that the product form solution does exist, then the breakthrough will extend the product form theory to networks with unbalanced flows; on the other hand, if it is shown that the product form solution does not exist in general, then one has to use some other techniques. An analogy to this would be that if it is shown that a problem can be solved with a polynomial time algorithm, then one can locate an optimal solution (an exact solution in the author's case); on the other hand, if it is shown that the problem is NP-complete, then one can only employ heuristic algorithms to solve the problem.

III.2 ASSUMPTIONS

It is useful to classify queueing networks before we investigate the existence problem for systems with unbalanced flows. Figure III.1 depicts all possible combinations of queueing networks as a big circle. The upper half of the big circle depicts networks with balanced flows and the lower half depicts networks with unbalanced flows.



(A) Networks with service time distributions which have rational Laplace transforms

(B) Networks with small population and/or few service facilities

(?) Networks corresponding to (A)

Figure III.1 Relationships among Queueing Networks

III.2.1 Networks with Balanced Flows

For networks with balanced flows (the upper half of the big circle), only a small number have exact solutions, as shown by the small circles (A) and (B). The small circle (A) stands for queueing networks which satisfy the assumption of the BCMP theorem and the small circle (B) stands for queueing networks with small population and/or few service facilities. It might be possible to find some other networks with balanced flows which have exact solutions. The point to emphasize here, though, is that, by and large, only a small percentage of networks with balanced flows have exact solutions. It is easy to construct networks with balanced flows which do not have known exact solutions. Examples are: queueing networks with FCFS service disciplines but with different service time distributions for different classes of customers; queueing networks with a moderate amount of service facilities and customers, 10 and 10 for instance, but with a finite buffer size, 20 for instance; queueing networks which allow re-routing; queueing networks which allow servers to idle when customers are in the queue; ; and queueing networks which have customers possessing more than one resource simultaneously. The list can go on and on.

For networks which have exact solutions, performance measures can be computed exactly and efficiently. If a network does not have an exact solution, then the analyst has to use either simulations or approximations which are more expensive and/or less accurate. Therefore, it pays to model a network with exact solutions. But there exists only a small percentage of queueing networks with balanced flows which can be analyzed exactly.

III.2.2 Networks with Unbalanced Flows

A network with unbalanced flows is one in which the input flow rate to a service facility (or a class of a service facility) may be different from its output flow rate. A formal definition of systems with unbalanced flows appears in Chapter IV. The question the author poses here is: under what kind of conditions may a network with unbalanced flows have an exact solution, specifically the product form solution described in the literature (5)?

A logical step to answering the question is to try to represent the state space of networks with unbalanced flows with a state-transition-rate diagram. Since a service time distribution with a rational Laplace transform has a stage representation (26, 27, 30) and the method of stages can be applied to construct state-transition-rate diagrams for networks with such service time distributions, it is logical to study networks corresponding to the small circle (A) in the upper half of the big circle where service time distributions are assumed to have Laplace Transforms. This kind of network is depicted by the small circle (?) in the lower half of the big circle. The other possibility is to investigate networks corresponding to the small circle (B) which have small population and/or few service facilities.

It is reasonable to argue that if one cannot find exact solutions for the network with unbalanced flows which correspond to the small circles (A) and/or (B), then it would be a formidable task to find exact solutions for other networks with unbalanced flows. On the other hand, if one can show that the product form solution does exist for some networks in the small circle (?) which may (or may not) have small population and/or few service facilities, then the results may be extended to more general networks. A moment of thought would lead one to try to solve for a special case in (?) with a small population and few service facilities. Chapter III.1.4 presents such a special case and discusses its implications.

III.2.3 Physical Characteristic

A more fundamental assumption has to be made before the author presents his approach to analyze the existence problem. It has been noted that the derivation of the BCMP theorem is based on the flow conservation argument and the input flow rate has to be equal to the output flow rate. A legitimate question to pose is how to apply the Markov state-transition-rate diagram to systems with unbalanced flows. The question is answered by assuming that flows do not have to be conserved at the flow unbalanced points -- an assumption which is consistent with the physical phenomenon observed in systems with unbalanced flows. Specifically, it is assumed that customers coming out of a service facility can split because of some physical phenomenon such as broadcast or acknowledgement. The effect of this assumption on the state-transition-rate diagram is discussed below.

Consider the state-transition-rate diagram of the BCMP type queueing network. If the network is flow balanced, then any two neighboring states in the state-transition-rate diagram can be expressed as follows:

before transition:

$(S_1, \ldots, S_i+(c), \ldots, S_j, \ldots, S_k-(c'), \ldots, S_M)$

after transition:

$(S_1, \ldots, S_i, \ldots, S_j, \ldots, S_k, \ldots, S_M)$

where S_j is a feasible state of service facility j,

$S_i+(c)$ is a feasible state with one more class c customer than state S_i,

$S_k-(c')$ is a feasible state with one less class c' customer than state S_k.

A transition from one state to another in a network with balanced flows can be interpreted as a customer finishing service at one facility and going to another facility. Whereas, if the network is flow unbalanced, then following the flow-unbalanced assumption discussed before, two neighboring states in the state-transition-rate diagram can be expressed as follows assuming that one customer has split into two customers before the transition occurs.

before transition:

$(S_1, \ldots, S_i+(c), \ldots, S_j-(c'), \ldots, S_k-(c''), \ldots, S_M)$

after transition:

$(S_1, \ldots, S_i, \ldots, S_j, \ldots, S_k, \ldots, S_M)$

This difference invalidates the proofs of the BCMP theorem, as discussed below. The key to the derivation of the product form solution for the BCMP type queueing networks with balanced flows is the concept of local balance. In a nutshell, it says that between any pair of states there should be either no transition at all or transitions should be in both directions and the rate in both directions should be equal (71). Chandy showed that if each service facility of a network satisfies local balance when isolated, then the equilibrium state probability density function of the network takes the product form solution (20). For BCMP type queueing networks with balanced flows, the local balance equation is satisfied. However, the BCMP theorem is not applicable to the BCMP type queueing network with unbalanced flows because, even though each service facility satisfies local balance when isolated, it is clear that a customer who finishes service at facility i does not simply go to another facility (or return to facility i for more service) if the customer is at a point with unbalanced flows. Instead, the customer splits into two (or more) customers and the two (or more) customers would go to two (or more) facilities in the network separately. It follows that in the proof of the BCMP theorem, one cannot apply the M => M (61) property to isolate a service facility from the rest of the facilities in the network, invalidating the theorem.

The author's experience indicates that it is very difficult, if not impossible, to try to work on the general form of a balanced equation in a network with unbalanced flows. Since the aim is to discover if the product form solution exists for systems with unbalanced flows, a simple case in the circle (?) is studied. Chapter III.3 elaborates on the approach and chapter III.4 works out such a case.

III.3 APPROACH

Two methods have been used in the literature (61, 71) to show whether the product form solution exists for a queueing network:

(I) Solve for the general balance equations and show that the steady-state joint probability distribution indeed is the product of individual service facility probability distributions;

(II) Let C be a normalization constant chosen such that the network state probabilities sum to one; and assume that

$$P(S_1, \ldots, S_M) = C\ P_1(S_1) \ldots P_M(S_M);$$

then check to see if consistent answers can be obtained from the general balance equations. If the results are consistent, then the product form solution satisfies the general equations; on the other hand, if contradictory results are derived, then the product form solution does not exist for the queueing network system in question. An example is given below to illustrate these two methods.

III.3.1 Example

Suppose that we have a closed system with only one customer and three service facilities. The service discipline of the facilities is FCFS, and the service time distribution is exponential. The routing probabilities are shown in Figure III.2, and the state-transition-rate diagram is shown in Figure III.3. Note that there is only one class of customers per service facility and flows are balanced. Therefore, the product form solution should exist in theory.

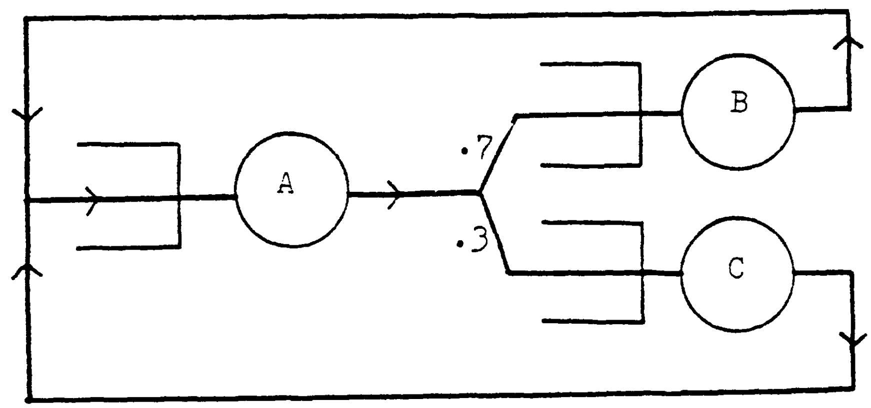


Figure III.2 Example of Queueing Network with Balanced Flows

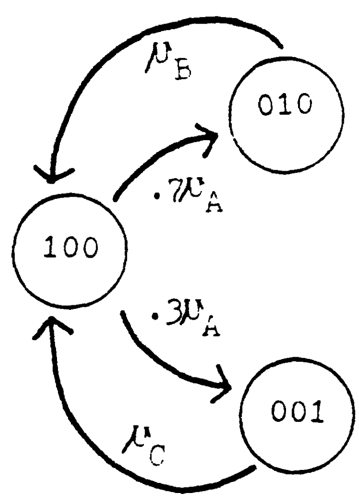


Figure III.3 State-Transition-Rate Diagram of Figure III.3.1

From the state-transition-rate diagram, one can derive the following balance equations:

$P(100) * 0.7\ \mu_A = P(010) * \mu_B$ ... (1)

$P(100) * 0.3\ \mu_A = P(001) * \mu_C$ ... (2)

$P(001) * \mu_C + P(010) * \mu_B = P(100) * \mu_A$ ... (3)

The two methods mentioned in III.1.3 are applied below to show that indeed for this flow balanced network, the product form solution exists.

III.3.2 Solution I: Solve for the General Balance Equations

The three general balance equations -- (1), (2), and (3) -- are solved below to show that the steady-state joint probability distribution has the product form solution.

From (1), $P(010)$

$= 0.7 * \mu_A / \mu_B * P(100)$ ... (4)

From (2), $P(001) = 0.3 * \mu_A / \mu_C * P(100)$ ... (5)

But $P(100) + P(001) + P(010) = 1$, therefore

$P(100) + 0.7 * \mu_A/\mu_B * P(100) + 0.3 * \mu_A/\mu_C * P(100) = 1$

It follows that, $P(100)$

$= 1 / (1 + 0.7 * \mu_A / \mu_B + 0.3 * \mu_A / \mu_C)$

$= (1/\mu_A) * 1/(1/\mu_A + 0.7/\mu_B + 0.3/\mu_C)$

Let $k = 1 / (1/\mu_A + 0.7/\mu_B + 0.3/\mu_C)$

It follows that, P(100)

$= k / \mu_A$

$= k * (1/\mu_A)^1 * (0.7/\mu_B)^0 * (0.3/\mu_C)^0$

P(010)

$= 0.7 * k / \mu_B$

$= k * (1/\mu_A)^0 * (0.7/\mu_B)^1 * (0.3/\mu_C)^0$

P(001)

$= 0.3 * k / \mu_C$

$= k * (1/\mu_A)^0 * (0.7/\mu_B)^0 * (0.3/\mu_C)^1$

But this is exactly the form shown by Gordon and Newell(35) which can be transformed to be the product of the probability distributions of the individual service facilities. Therefore, the p.f.s. does exist.

III.3.3 Solution II: Assume the Product Form Solution Exists

Assume that $P(S_1, \ldots, S_M) = C\ P_1(S_1) \ldots P_M(S_M)$, then

From (1), $P_A(1)\ P_B(0)\ P_C(0) * 0.7\ \mu_A$

$= P_A(0)\ P_B(1)\ P_C(0) * \mu_B$

Therefore, $P_A(1)\ P_B(0) * 0.7\ \mu_A$

$= P_A(0)\ P_B(1) * \mu_B$ ... (4)'

From (2), $P_A(1)\ P_B(0)\ P_C(0) * 0.3\ \mu_A$

$= P_A(0)\ P_B(0)\ P_C(1) * \mu_C$

Therefore, $P_A(1)\ P_C(0) * 0.3\ \mu_A$

$= P_A(0)\ P_C(1) * \mu_C$ ... (5)'

From (3), $P_A(0)\ P_B(0)\ P_C(1) * \mu_C + P_A(0)\ P_B(1)\ P_C(0) * \mu_B$

$= P_A(1)\ P_B(0)\ P_C(0) * \mu_A$

Plug (4)' and (5)' to the left hand side above,

it follows that the left hand side

$= P_A(1)*P_C(0)*0.3*\mu_A*P_B(0) + P_A(1)*P_B(0)*0.7*\mu_A*P_C(0)$

$= P_A(1)*P_B(0)*P_C(0)*\mu_A$

= the right hand side.

That is, all the above balance equations hold when the product form solution is used to verify the results. It is ideal to show that the product form solution exists by method (I), but in general it is difficult because the number of general balance equations explodes as the population or the number of service facilities of the system increases. Method (II) is employed in the next section to study systems with unbalanced flows.

III.3.4 Case Study

A case is examined in this section to see if the product form solution can exist for systems with unbalanced flows. The queueing network diagram for the case is shown in Figure III.4. Note that the routing probabilities from facility A to both facility B and facility C equal to one, a violation of the flow balanced assumption used by classical queueing networks. Assuming that customers coming out of a service facility can split, then the corresponding state-transition-rate diagram for Figure III.4 can be derived as shown in Figure III.5.

From the state-transition-rate diagram, we get

μ_C P(011) = μ_B P(010) ... (1)

μ_A P(100) = μ_B P(010) + μ_C P(101) ... (2)

(μ_A + μ_C) * P(10,I)
= μ_B P(01,I) + μ_C P(10,I+1) ... (3)
for I = 1, 2, ...

(μ_B+μ_C) * P(01,I) = μ_C*P(01,I+1) + μ_A* P(10,I-1) ... (4)
for I = 1, 2, ...

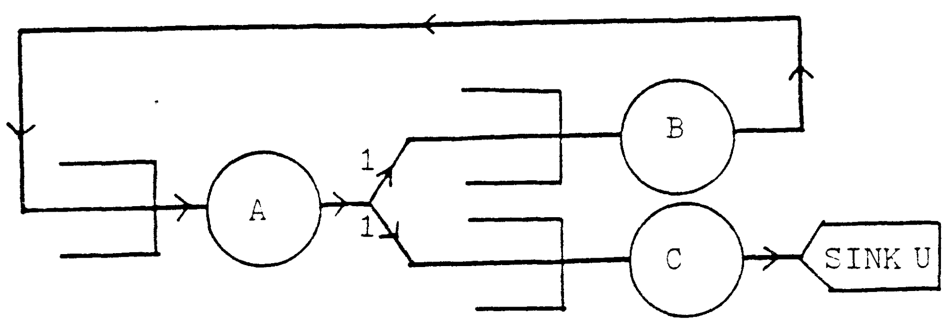


Figure III.4 Example of Queueing Network with Unbalanced Flow

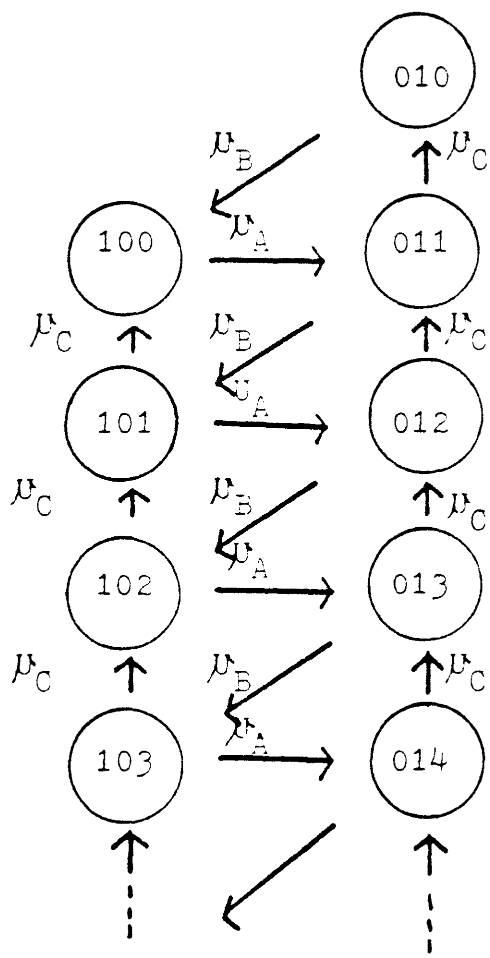


Figure III.5: State-Transition-Rate Diagram of Figure III.4.1

Suppose that $P(S_A, S_B, S_C) = C\ P_A(S_A) * P_B(S_B) * P_C(S_C)$

Then from (1), $\mu_C\ P(011)$

$= \mu_C * C * P_A(0) * P_B(1) * P_C(1)$

$= \mu_B * C * P_A(0) * P_B(1) * P_C(0)$

It follows that, $\mu_C * P_C(1) = \mu_B * P_C(0)$... (5)

From (2), $\mu_A * P_A(1) * P_B(0) * P_C(0)$

$= \mu_B * P_A(0) * P_B(1) * P_C(0) + \mu_C * P_A(1) * P_B(0) * P_C(1)$

$= \mu_B * P_A(0) * P_B(1) * P_C(0) + \mu_B * P_C(0) * P_A(1) * P_B(0)$

It follows that, $\mu_A * P_A(1) * P_B(0)$

$= \mu_B * P_A(0) * P_B(1) + \mu_B * P_A(1) * P_B(0)$

Therefore, $(\mu_A - \mu_B) * P_A(1) * P_B(0) = \mu_B * P_A(0) * P_B(1)$ ... (6)

From (3), $(\mu_A + \mu_C) * P_A(1) * P_B(0) * P_C(I)$

$= \mu_B * P_A(0) * P_B(1) * P_C(I) + \mu_C * P_A(1) * P_B(0) * P_C(I+1)$

for I = 1, 2, ...

Plug (6) into the above equation, it follows that,

$(\mu_A + \mu_C) * P_A(1) * P_B(0) * P_C(I)$

$= (\mu_A - \mu_B) * P_A(1) * P_B(0) * P_C(I) + \mu_C * P_A(1) * P_B(0) * P_C(I+1)$

for I = 1, 2, ...

It follows that, $(\mu_A + \mu_C) * P_C(I)$

$= (\mu_A - \mu_B) * P_C(I) + \mu_C * P_C(I+1)$

for I = 1, 2, ...

i.e. $P_C(I+1) = (\mu_B + \mu_C) / \mu_C * P_C(I)$

for I = 1, 2, ...

i.e. $P_C(1) < P_C(2) < P_C(3) < \ldots$

Contradictory to the fact that,

$P_C(0) + P_C(1) + P_C(2) + \ldots = 1$

Therefore, the product form solution does not hold in this case. In other words, a counter example has been identified for systems with unbalanced flows. That is, exact solutions do not exist in general for systems with unbalanced flows with the assumptions made in this chapter. A cutting technique is presented in the next chapter to model and analyze distributed systems with unbalanced flows.

CHAPTER IV

Modeling and Analysis of Distributed Systems with Unbalanced Flows

It was shown in Chapter III that the product form solution does not exist in general and other approaches such as approximations have to be applied. A model and a cutting technique is presented in this chapter to model distributed systems with unbalanced flows. Issues and solutions derived from the cutting technique are discussed.

IV.1 MODEL STRUCTURE

Without loss of generality, let's assume that all customers in the queueing network are homogenous, i.e. there is a single customer type. In Figure II.1, the single type customer has 0.8 probability of requesting the read operation and 0.2 probability of requesting the write operation. It would be easy to relax this assumption to include different types of customers.

Let there be M service facilities and C classes in a queueing network. A service facility may consist of several classes which allow customers to have different sets of routing probabilities for different visits. Assume that any sources and sinks belong to class 0. Let $p_{i,j}$ denote the routing probability which is the fraction of the customers completing service in class i that joins class j. $i = 0, \ldots, C$, $j = 0, \ldots, C$; $p_{0,0} = 0$ by convention.

A main chain is defined as the path through which customers travel according to the defined routing probability and eventually go out of the system to return to the reference source. Since all customers have been assumed to be homogeneous, there is only one main chain in the system. In Figure II.2 the classes (SOURCEM, RP1, RD1, WP1, SINKM) define the main chain.

A class c customer of facility m in the queueing network is said to be UAP with degree b ,i.e. UAP(c,m)=b, if its output splits into b branches where b is a real number greater than one but each branch has a routing probability not greater than one. In Figure II.2, UAP(WP1,P1) = 2. Note that (a) UAP can occur in many classes within a queueing network; for instance, acknowledgements may take place at different levels of a data storage hierarchy; and (b) the inputs to a class that cause UAP can be the outputs from other UAP classes. For instance, a split from an acknowledgement may split again to send more acknowledgements to other classes.

Consider a class which is UAP with degree b. The main task that eventually returns to the reference source is defined as belonging to the main chain; on the other hand, the b-1 additional flows which cause that class to be unbalanced are perceived as "internal sources" (denoted as SOURCEU) which generate customers to travel within the network and eventually terminate at the "internal sink" (denoted as SINKU). It follows, as will be justified in Chapter IV.2, that all the classes with

UAP can be separated from the main chain to form the UAP chain where the <u>UAP chain</u> is defined as the additional path through which the "internally generated" customers (from SOURCEU) travel and eventually sink (at SINKU). In Figure II.3, the classes (SOURCEU, WD1, WD2, SINKU) define the UAP chain. Note that SOURCEU may stand for multiple "internal sources".

By labeling (source,sink) of the main chain as (SOURCEM, SINKM) and others as (SOURCEU, SINKU), one can decompose the graph of a network model with UAP unambiguously without referring to the semantics of the model. In other words, given the labeled graph of an UAP network, it is impossible to interchange one of the UAP flows with a part of the main chain. Therefore, a unique syntactic definition exists for each UAP network.

Classical queueing network models cannot be applied to analyze UAP directly because of the unbalanced flows mentioned. An extended routing matrix is introduced below to accommodate the problem.

Let R denote the extended routing matrix of an UAP network where a row-sum may be greater than one. The extended routing matrix R for Figure II.1 is shown in Figure IV.1.

Let R_c denote the unextended routing matrix which excludes the UAP chain of the network. The unextended routing matrix R_c which excludes the UAP chain (SOURCEU, WD1, WD2, SINKU) is shown in Figure IV.2. Elements in R and R_c are the routing probabilities $p_{i,j}$'s.

Define the visit ratio of a class, V_c, as the mean number of requests of the class to a service facility per customer. Define the sum of visit ratios of all exogenous sources, V_o, in an open system to be one. In a closed system, the outputs feedback to the system inputs; the sum of visit ratios of the system inputs is also defined to be one.

The visit ratios of the classes in R_c can be obtained from the visit ratio equations (6, p.237), viz.,

$$V_j = p_{o,j} + \sum_{i=1}^{C} V_i * p_{i,j} \qquad j = 1, \ldots, C.$$

$$R = \begin{array}{c|ccccccc} & RP1 & WP1 & RD1 & SINKM & WD1 & WD2 & SINKU \\ \hline SOURCEM & & & & & & & \\ RP1 & & & & & & & \\ RD1 & & & & & & & \\ WP1 & & & & & & & \\ WD1 & & & & & & & \\ WD2 & & & & & & & \end{array}$$

$$R = \begin{array}{l} \\ SOURCEM \\ RP1 \\ RD1 \\ WP1 \\ WD1 \\ WD2 \end{array} \begin{array}{c} \begin{array}{ccccccc} RP1 & WP1 & RD1 & SINKM & WD1 & WD2 & SINKU \end{array} \\ \begin{bmatrix} .8 & .2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

Figure IV.1:
<u>The Extended Routing Matrix for Figure II.1</u>

$$R_c = \begin{array}{l} \\ SOURCEM \\ RP1 \\ RD1 \\ WP1 \end{array} \begin{array}{c} \begin{array}{cccc} RP1 & WP1 & RD1 & SINKM \end{array} \\ \begin{bmatrix} .8 & .2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

Figure IV.2:
<u>The Unextended Routing Matrix for Figure II.1</u>

The visit ratios of classes in the UAP chain can be obtained once the visit ratios of the classes in the main chain are known. In Figure IV.2, let the visit ratio of class SOURCEM be 1 (recall the sum of visit ratios of all exogenous sources is defined to be one), and let the indices for (SOURCEM,RP1,WP1,RD1,SINKM) be (0,1,2,3,0), then

$p_{0,1} = 0.8$; $p_{0,2} = 0.2$; $p_{0,3} = 0$; $p_{0,0} = 0$.

$p_{1,1} = 0$; $p_{1,2} = 0$; $p_{1,3} = 1$; $p_{1,0} = 0$.

$p_{3,1} = 0$; $p_{3,2} = 0$; $p_{3,3} = 0$; $p_{3,0} = 1$.

$p_{2,1} = 0$; $p_{2,2} = 0$; $p_{2,3} = 0$; $p_{2,0} = 1$.

$\Rightarrow V_1 = 0.8$; $V_2 = 0.2$; $V_3 = V_1 = 0.8$

i.e. the visit ratios of (SOURCEM,RP1,WP1,RD1,SINKM) = (1,.8,.2,.8,1). Since SOURCEU has the same visit ratio as WP1 which is 0.2, it follows that SOURCEU = 0.2; and (SOURCEU,WD1,WD2,SINKU) = (.2,.2,.2,.2)

Alternatively, the visit ratio equations can be applied directly to the extended routing matrix R to obtain all the visit ratios of the classes in R.

IV.2 ANALYTIC FORMULATION OF QUEUEING NETWORKS WITH UAP

It was noted, in Chapter IV.1, that a) UAP can occur in many classes within a queueing network; that b) an input to a class that causes UAP may be the output from another UAP class; and that c) all the additional unbalanced flows are defined as belonging to the UAP chain -- a single chain. It is natural to ask whether the flows of the transformed network would be balanced, and what kind of relationship would exist between the main chain and the UAP chain. These questions are answered below:

If one cuts the additional b-1 unbalanced flows from a class which is UAP with degree b and inserts "internal sources" (SOURCEU) which generate customers with equivalent flow rates as those of the network before the cut, then following the assumption that unbalanced flows run independently of one another except for resource contention, the b-1 unbalanced flows will form b-1 new open chains which will not interact with the main chain. If all the additional unbalanced flows (spawned from the classes which are UAP and connected to the main chain) are cut from the main chain, then the flow in the main chain will be balanced, as illustrated in Figure II.2.

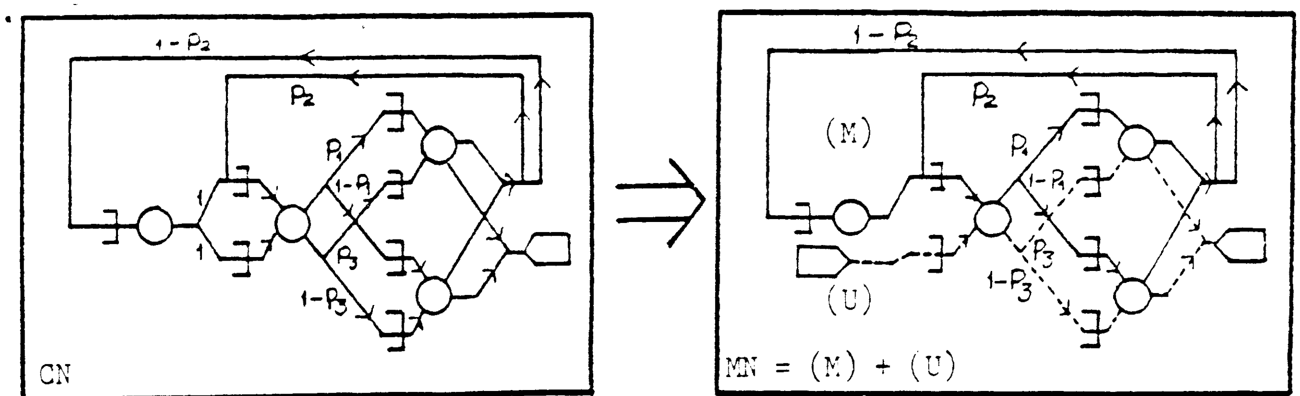
Let {R} denote the set of classes in the network before the cuts and $\{R_c\}$ denote the set of classes in the main chain, as illustrated in Figure IV.1 and IV.2. It follows that we have the balanced main chain with its classes in the set $\{R_c\}$ and many open chains with their classes in the set $\{R\} - \{R_c\}$. Therefore, the classes in the main chain and the classes in the open chains are disjoint.

However, it has been pointed out in Chapter IV.1 that a split may split again, so the open chains may themselves be flow unbalanced. To solve the problem, it is logical to cut all the additional unbalanced flows in the open chains continuously (and insert "internal sources" which generate equivalent flow rates as those of the open chains before the cuts) until all flows are balanced, forming very many open chains.

It is assumed that service time distributions and service disciplines of the facilities in the network follow those of Chapter II.3; in addition, the unbalanced flows which run independently of one another are assumed to arrive at their destinations as independent Poisson processes (this assumption is also adopted by other researchers (34, 39)). The simulation studies the author has conducted indicate that this assumption is fairly robust. The validation reported by Goldberg, Popek, and Lavenberg (32) provide further support for this assumption. It follows that the OPFMCQN result can be applied to aggregate the very many open chains discussed in the last paragraph to a single open chain -- the UAP chain.

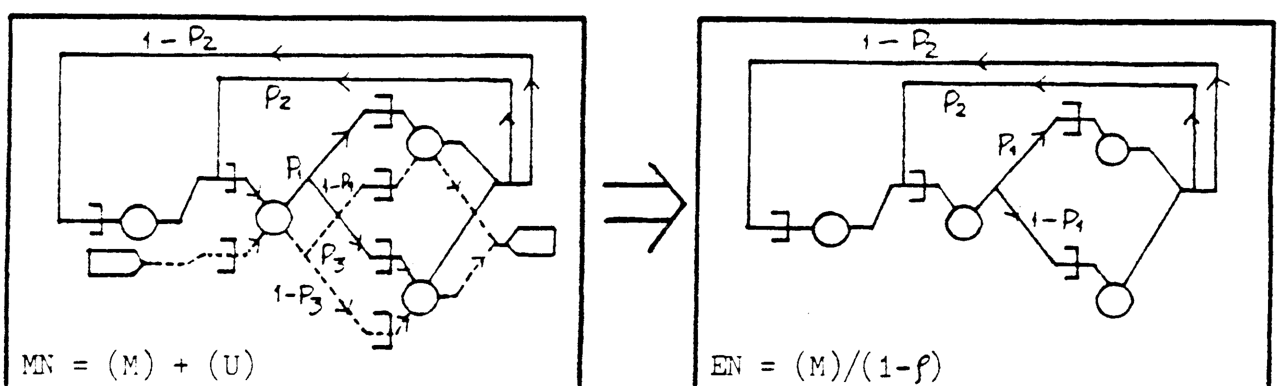
If the original network is an open network, then the OPFMCQN result can be applied again to make the overall network a single chain with its workload contributed from both the main chain and the UAP chain. Chapter IV.2.1 discusses the formulation of useful performance measures for open queueing networks with UAP. On the other hand, if the original network is a closed network, then we have a mixed network with the closed main chain and the open UAP chain, as illustrated in Figure IV.3; Chapter IV.2.2 discusses the necessary and sufficient condition for the closed network to be stable and an iterative procedure which computes the system throughput.

It is extricable now to formulate networks with UAP. Let the summation of visit ratios over all the cuts, V(U), denote the "internally generated" visit rate of the UAP chain. Note that "(M)" will denote an open chain in Chapter IV.2.1 and a closed chain in Chapter IV.2.2.



Original Closed Network with UAP(CN)   Mixed Network without UAP(MN)

Figure IV.3: Decomposition of CN to MN



Mixed Network without UAP(MN)   Equivalent Network without UAP(EN)

Figure IV.4: Transformaion of MN to EN for the Main Chain

IV.2.1 Open Queueing Networks with UAP

For an open queueing network with UAP, the network arrival process is assumed to be Poisson with a constant rate λ_o. By solving the extended routing matrix introduced in Chapter IV.1, one can obtain the visit ratios for all classes, hence V(U). Since λ_o is given, $X_o(U)$ is also determined, specifically, $X_o(M) = \lambda_o$ and $X_o(U) = \lambda_o * V(U)$. For instance, suppose $\lambda_o = 5$ customers/sec in Figure II.1, then the UAP chain (SOURCEU, WD1, WD2, SINKU), as shown in Figure II.3, has an arrival rate of 1 customer/sec.

Since the network can be aggregated to an open single chain network, its stability follows from OPFSCQN, i.e. the network is stable if and only if $U_i < 1$ for all facilities in the network. It can be shown (49, 71) that throughput, utilization, mean queue length, and response time are computed as shown in Table IV.1. Note that:

I). The denominator of $N_i(M)$ is U_i which quantifies the resource contention between the UAP chain and the main chain.

II). $R_o(M)$ is the "system response time" the reference source perceives instead of R_o.

III). $X_i(M)$ would be the sum of the products of visit-ratios and mean service times if there were multiple classes of customers at facility i for the main chain; the same situation happens to the UAP chain.

Facility i	FCFS,PS,LCFSPR discipline
$X_i(M)$	$X_o(M) * V_i(M)$
$X_i(U)$	$X_o(U) * V_i(U) / V(U)$
X_i	$X_i(M) + X_i(U)$
$U_i(M)$	$X_i(M) * S_i(M)$
$U_i(U)$	$X_i(U) * S_i(U)$
U_i	$U_i(M) + U_i(U)$
$N_i(M)$	$U_i(M) / (1-U_i)$
$N_i(U)$	$U_i(U) / (1-U_i)$
N_i	$N_i(M) + N_i(U)$
$R_i(M)$	$N_i(M) / X_i(M)$
$R_i(U)$	$N_i(U) / X_i(U)$
R_i	$N_i(M) + N_i(U)$
$R_o(M)$	$R_1(M) + \ldots + R_C(M)$
R_o	$R_1 + \ldots + R_C$

Table IV.1:
Formulae for Open Queueing Networks with UAP.

IV.2.2 Closed Queueing Networks with UAP

For closed queueing networks with UAP, a mixed network with the closed main chain and the open UAP chain, as illustrated in Figure IV.3 can be obtained following the discussion in Chapter IV.2. Since $X_o(U) = X_o(M) * V(U)$ where $X_o(M)$ is evaluated through a nonlinear function of $X_o(U)$ (Chapter II.3), it follows that $X_o(U) = f(X_o(U)) * V(U)$ where f is a nonlinear function. To solve the nonlinear equation, two issues have to be addressed first:

A) What are the properties of f?

B) What is the necessary and sufficient condition for the network to be stable?

A corollary based on Reiser and Kobayashi's theorem (64) on PFMQN is shown below to settle issue A; and two lemmas are proven to settle issue B which leads to an iterative procedure for the closed network. The IS discipline is excluded from this subsection, Chapter IV.2.3 discusses its difference from other disciplines.

A) **Corollary:** An equivalent closed network (EN) of the main chain for the mixed network (MN), as illustrated in Figure IV.4, can be obtained by inflating the main chain traffic intensities, i.e. by replacing $\rho_i(M)$ by $\rho_i(M) / (1-\rho_i(U))$ for i = 1, ..., M.

Proof: Define(64) the p.g.f. for

$P(n_1(M), n_1(U), \ldots, n_M(M), n_M(U))$ as

$$G(Z,\theta) = \Pi\ \Phi_i(\ \rho_i(U) * z_i(U) + \rho_i(M) * z_i(M) * \theta)$$

where z_i is the p.g.f. transformation variable for facility i; θ is a factor associated with the main chain to insure that main chain population is fixed to N; the product, Π, is taken from 1 up to M, and $\Phi_i(\zeta) = 1/(1-\zeta)$ for FCFS, PS, and LCFSPR. The p.g.f. is found as the coefficient of θ^N in a power series expansion of $G(Z,\theta)$ in θ, denote it $G^*(Z)$. It follows that

$$G^*(Z) = C * \partial_\theta(N) * \Pi\ \Phi_i(\rho_i(U) * z_1(U) + \rho_i(M) * z_i(M) * \theta)$$

To obtain the p.g.f. of the marginal distribution of the closed main chain, let $z_i(U)=1$. It follows that

$$G^*\ (z_i(U)=1)$$
$$= C * \partial_\theta(N) * \Pi\ \Phi_i\ (\rho_i(U) + \rho_i(M) * z_i(M) * \theta)$$
$$= C * \partial_\theta(N) * \Pi\ 1/(\ 1 - \rho_i(U) - \rho_i(M) * z_i(M) * \theta\ )$$
$$= C * (\Pi\ 1/(1 - \rho_i(U))) * \partial_\theta(N)$$
$$* \Pi\ 1/(1-(\ \rho_i(M) * z_i(M) * \theta\ /\ (\ 1 - \rho_i(U))))$$
$$= (1/G(N)) * (\ \Sigma\Pi\ (\rho_i(M) * z_i(M)\ /(\ 1 - \rho_i(U)))^{ni(M)})$$

where the summation is taken over all possible states of $S(N,M) = \{\ (n_1(M), \ldots, n_M(M)) |\ n_1(M) + \ldots + n_M(M) = N$, and $n_i(M) \geq 0$ for all $i\ \}$. But this is exactly the p.g.f. for CPFSCQN with the traffic intensity inflated by $(1-\rho_i(U))^{-1}$ for facility i. Q.E.D.

From the marginal distribution above, it is not difficult to show (39) that f is CMD, assuming that there exists at least a pair of $(D_i(M), D_i(U))$ such that $D_i(M)>0$ and $D_i(U)>0$. With the

corollary and the CMD property, the convolution algorithm can be applied to solve the nonlinear equation iteratively. Let $()^i$ denote the ith iteration. For instance, $(EN(X_o))^{10}$ denotes the throughput of EN at the 10th iteration. $(X_o(U))^0$ is given initially. $(X_o(U))^{i+1}$ is estimated as follows:
$(X_o(U))^{i+1} = (EN(X_o))^{i+1} * V(U)$ and $(EN(X_o))^{i+1} = f((X_o(U))^i)$.
This relationship is used below.

B) Since the stability of PFMQN is unaffected by the presence of closed chains (Chapter II.3), it follows that a closed network with UAP is stable if and only if MAX $u_i(U)<1$ where $i = 1, ..., M$, and $u_i(U) = (X_o(U) / V(U)) * D_i(U)$.

Denote MAX $D_i(U)$ as $D_I(U)$, and denote $V(U)/D_I(U)$ as B; then it follows that a closed queueing network with UAP is stable if and only if $X_o(U) < B$.

Denote $D_I(M)$ as the main chain D value at facility I; then the stability condition of the closed network with UAP can be identified with the following four mutually exclusive and collectively exhaustive cases:

I) $f(X_o(U)=0) * V(U) < B$.

II) $f(X_o(U)=0) * V(U) \geq B$, but $D_I(M) > 0$.

III) $f(X_o(U)=0) * V(U) \geq B$, $D_I(M) = 0$, but $f(X_o(U)=B) * V(U) < B$.

IV) $f(X_o(U)=0) * V(U) \geq B$, $D_I(M) = 0$, and $f(X_o(U)=B) * V(U) \geq B$.

Figure IV.5 depicts the four conditions and the lemma below establishes the condition for stability.

Let $a = f(X_o(U)=0)$, $b = a*V(U)$, $c = f(X_o(U)=B)$, and $d = c*V(U)$; then the four cases can be rewritten as follows:

I) $b < B$.

II) $b \geq B$, but $D_I(M) > 0$.

III) $b \geq B$, $D_I(M) = 0$, but $d < B$.

IV) $b \geq B$, $D_I(M) = 0$, and $d \geq B$.

Lemma: The network is stable if and only if it is not case IV.

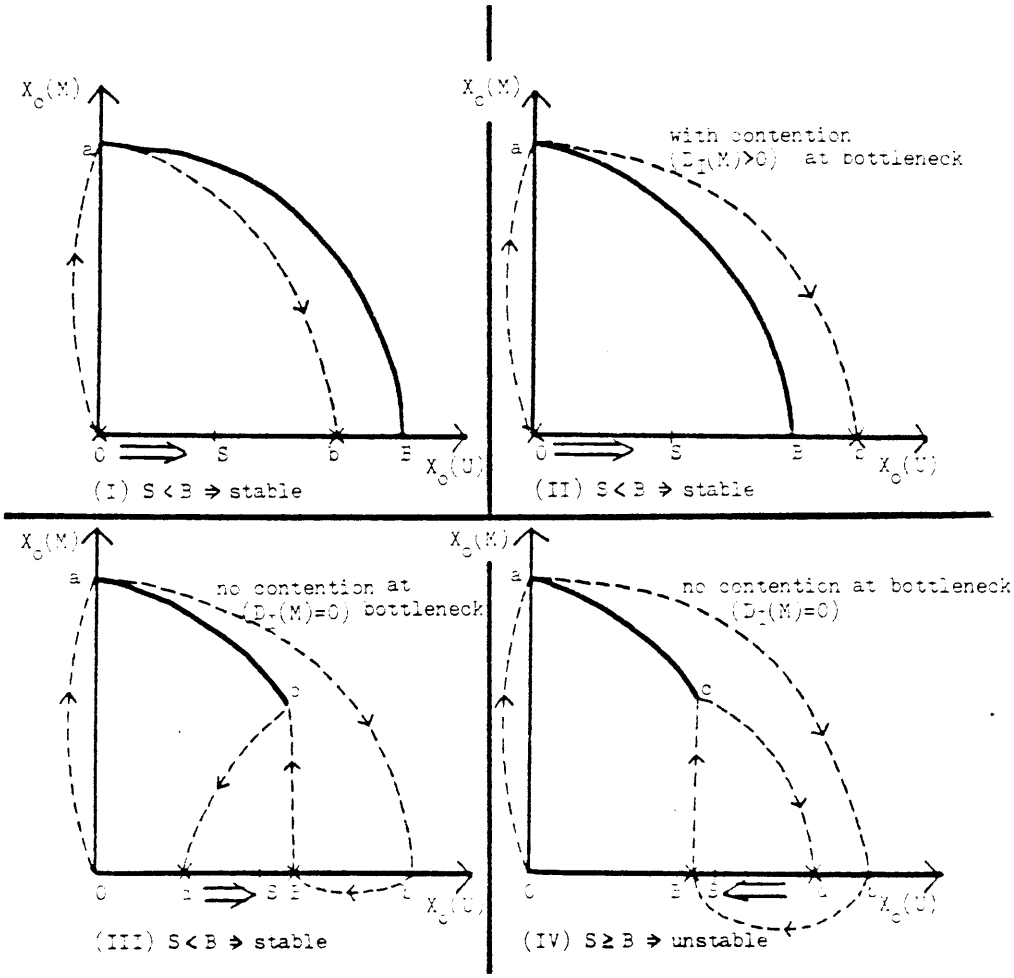
Proof: Case I states that zero is given as the initial estimate for $(X_o(U))^0$, and $(X_o(U))^1 = (EN(X_o))^1 * V(U) = b < B$, as shown in Figure IV.5.I. Since f is CMD and a is the upper bound of the main chain throughput, it follows that $(X_o(U))^i$ is bounded between 0 and b for all i. Therefore, the stability condition is held since $b < B$.

Case II states that zero is given as the initial estimate for $(X_o(U))^0$, and $(X_o(U))^1 \geq B$ as shown in Figure IV.5.II, but there exists contention at the bottleneck facility I. Suppose a solution exists between B and b, i.e. $B \leq (X_o(U))^\infty = (EN(X_o))^\infty * V(U) \leq b$. It follows that $(EN(X_o))^\infty \geq B/V(U) > 0$. On the other hand, there exists contention at facility I, therefore $(EN(X_o))^\infty = 0$ because the bottleneck facility I is fully utilized by the open UAP chain, blocking the closed main chain flow completely.

However, this is contradictory to the supposition; therefore, the solution is bounded in the open interval (0,B) which is less than B and the condition is held.

Case III states that there is no contention at the bottleneck facility. B is given as the initial estimate for $(X_o(U))^0$, and $(X_o(U))^1 = d < B$ as shown in Figure IV.5.III. It follows, by CMD, that a solution exists in the open interval (d,B) and the condition is held. Note that $D_I(M) = 0$ implies that the bottleneck facility I does not contribute to the main chain throughput at all. The only impact it has is to cause the overall network to be unstable.

Case IV states that there is no contention at the bottleneck facility and $(X_o(U))^1 = d \geq B$. It follows, by CMD, that if a solution exists, it must be greater than or equal to B, violating the stability condition. Q.E.D.



Key

$a = f(X_o(U)=0)$ $\quad$ $b = f(X_o(U)=0)*V(U)$

$c = f(X_o(U)=B)$ $\quad$ $d = f(X_o(U)=B)*V(U)$

B: bottleneck throughput

S: solution

$X_o(M)$: Main chain throughput $\quad$ $\times\!\!-\!\!\times$: bound

$X_o(U)$: UAP chain throughput

Figure IV.5: Four Cases to Test the Stability Condition

Several important insights are summarized below:

a) Case II occurs when the external workload (the main chain) and the internal overhead (the UAP chain) contend for the bottleneck facility. A good design would balance the contention according to the traffic intensities or take advantage of case III.

b) Case III can be used to design systems with higher throughput by offloading UAP to a separate processor which does not contend any resource with the main chain. Consider the throughput a manager would gain if he could offload all but the critical task to his assistants who would finish the assigned tasks independently without bothering the manager at all.

c) Case IV is not uncommon: consider a bad architectural design where too many unbalanced flows are directed to some specialized hardware for table-update; if the specialized hardware is slow by design to reduce cost, then it is likely that the system will be unstable. Erroneous design decisions can be reduced by excluding this possibility.

d) The equilibrium condition,if it exists, is unique because f is CMD.

e) The stability condition can be insured by excluding case IV.

f) The convolution algorithm, simple and efficient, is used to insure the stability condition as well as to locate the solution.

g) The equivalent closed network obtained from the corollary is used to calculate the "system response time" perceived by the reference source. Moreover, when the iterative procedure stops, G(1), ..., G(N) are also available as a by-product for calculating useful performance measures.

IV.2.3 Discussion

An analytic technique has been developed to model distributed systems with unbalanced flows due to asynchronously spawned tasks (UAP). Assumptions have been made without loss of generality to focus the presentation on the UAP phenomenon. It would be easy to relax the fixed service rate to include the load dependent service rate. The IS discipline was excluded in Chapter IV.2.2 since the main chain and the UAP chain do not interact with each other at the IS facility. For networks with mixed disciplines, the inflating factor for the IS facility is one. For networks with IS facilities only, the UAP chain has no impact on the main chain, therefore, can be ignored.

IV.3 PRIORITY SCHEDULING OF DISTRIBUTED SYSTEMS WITH UNBALANCED FLOWS

Distributed systems with unbalanced flows have been modeled and analyzed in Chapter IV.1 and IV.2 for a broad range of queueing network models including pragmatic features of computer systems such as distinct classes of jobs, general service time distributions, and scheduling disciplines such as FCFS, LCFSPR, and PS. However, the priority scheduling discipline has not been modeled because it does not satisfy the constraints that guarantee the product form solution even in models with balanced flows.

The advantages of priority scheduling in computer systems, for higher performance and better resource utilization, make it highly desirable to model the priority scheduling discipline for systems with unbalanced flows. To illustrate the practicality of priority scheduling, let's consider the transactions that support the read and write requests in the INFOPLEX data storage hierarchy (46).

It would be ideal to process read requests as soon as possible so that the response time that the reference source perceives can be minimized. By the same token, it is desirable to return an acknowledgement to a write request as soon as the data to be updated is committed. On the other hand, since transactions such as the STORE-BEHIND operations are transparent to the reference source, they can be processed at a later time as long as it is guaranteed that the data will be updated at the lower levels of the data storage hierarchy. Thus, the STORE-BEHIND operations at the lower levels of the data storage hierarchy can be assigned a lower priority. As a result, the response time to the external users for read and write requests will be enhanced.

IV.3.1 Techniques for Flow Balanced Systems

Techniques for studying priority scheduling disciplines in queueing network models have been proposed (49, 80). Sevcik (80) proposed the "shadow CPU" technique to approximate a central server model with the preemptive priority scheduling discipline at the CPU and FCFS at the I/O channels. Basically, his approach is as follows: suppose there are two types of customers visiting the CPU, one with a higher prioirty and the other with a lower priority. To eliminate the CPU contention due to the higher prioirty customers, an additional CPU (called the "shadow CPU") is provided for the exclusive use of the lower priority customers. Clearly the lower priority customers will be receiving unrealistically good service at the CPU because they don't contend with the higher priority customers. Therefore, the lower priority customers will congest the I/O channels more than they actually would in the priority scheduling model. A variation of the "shadow CPU" model involves slowing down the progress of the lower priority customers by reducing the service rate of the "shadow CPU" to reflect the CPU utilization by the higher priority customers. This is be done by multiplying the lower priority customer's mean service time at the shadow CPU by $1/(1-U_H)$ where U_H is the utilization of the CPU by the higher priority customers. While U_H is not known a priori in a closed system, a binary search can be used to determine the self-consistent utilization (80).

For a distributed system where the lower priority customers may travel through a set of service facilities, a generalized queueing model instead of a central server model has to be employed. To reflect the contention due to the higher priority customers, the service rates of the lower priority customers should be reduced by $1/(1-U_{Hi})$ where U_{Hi} is the utilization of facility i due to the higher priority customers.

The techniques mentioned in this section are useful conceptually in developing techniques for systems with unbalanced flows which are presented in the next section.

IV.3.2 Techniques for Flow Unbalanced Systems

It is assumed that the distributed systems with unbalanced flows have a preemptive priority in favor of the main chain. Moreover, it is assumed that some of the additional unbalanced flows such as those due to the STORE-BEHIND operations have a lower priority while others have the same priority as the main chain. Let the preemptive priority customers be called type H customers and the lower priority customers be called type L customers. To reflect the contention due to type H customers, type L customers have to be slowed down. However, the response time of type L customers is irrelevant to the response time that the reference source perceives because type L customers are fully preempted. In other words, type L customers are transparent the the external world. Therefore, it is unnecessary to adjust the service rate of type L customers unless one became interested in the response time of type L customers.

To compute the performance measures of systems with unbalanced flows with different priorities, as assumed before, one simply ignores type L customers in calculating the sum of the products of visit ratios and mean service times. However, the stability condition has to be checked with type L customers included. Otherwise, the system may become unstable due to excessive backlog of type L customers.

Distributed systems with unbalanced flows and with different priorities have been modeled. However, the model is restricted to the case where some of the unbalanced flows have a lower priority than the main chain. Conceivably, it would be more complicated if some of the unbalanced flows require a higher priority than the main chain. This kind of systems remains to be studied. An optimistic bound of the approximation can be easily obtained by ignoring the lower priority customers completely, while a pessimistic bound can be obtained by assuming that all customers have the same priority (i.e. with the PS discipline).

The theory developed in Chapter IV.2 was implemented in a software package called TAD (Technique for Architectural Design) which is presented in Chapter V. Simulation results are presented in Chapter VI to validate the techniques.

CHAPTER V

Efficiency of Iterative Algorithms and Implementation of TAD

The theory developed in Chapter IV.2 was investigated further to study its applicability. Two iterative algorithms were studied to compare their converging speeds. The results of the study were implemented in TAD to evaluate the performance of different design alternatives of the INFOPLEX data storage hierarchy. The efficiency of the two algorithms and the implementation of TAD are presented in this chapter to demonstrate the practicality of this research.

V.1 ITERATIVE ALGORITHMS

It was shown in Chapter IV.2 that the stability condition of a closed system can be identified to insure that a unique equilibrium system throughput, X_o, exists. To locate X_o, Buzen's convolution algorithm, as shown in Algorithm V.1, is applied to solve the nonlinear equation, G(N-1)/G(N), iteratively, where G(N) is the normalization constant when N customers circulate in the closed system. The computational efficiency of each iteration is the order of M*N(o(MN)) where M is the number of service facilities (14). In practice, it is common to have a closed system with 10 customers and 15 service facilities. For instance, a P1L3 INFOPLEX data storage hierarchy model with 10 degrees of multiprogramming may be represented as a closed system with 10 customers and 15 service facilities. In this case, it

```
REM ========== CONVOLUTION.ALGORITHM ============

FOR M=1 TO NUMBER.OF.FACILITIES
|   IF
|      VSM(M)>0
|         THEN
|            INFLATED.VSM(M) = VSM(M)/(1-VSU(M)*X.EST)
|         ELSE
|            INFLATED.VSM(M) = 0
NEXT M
FOR N = 1 TO NUMBER.OF.CUSTOMERS
|   G(N)=0
NEXT N
G(0) = 1
FOR M = 1 TO NUMBER.OF.FACILITIES
|   FOR N=1 TO NUMBER.OF.CUSTOMERS
|   |   G(N)=G(N)+INFLATED.VSM(M)*G(N-1)
|   NEXT N
NEXT M
XM =G(NUMBER.OF.CUSTOMERS-1)/G(NUMBER.OF.CUSTOMERS)
RETURN
```

Algorithm V.1:
The Inflated Convolution Algorithm

would take approximately 150 additions and 150 multiplications for each iteration. As the number of customers and the number of service facilities increase, (for instance, a P5L4 data storage hierarchy model with 20 degrees of multiprogramming may be represented as a closed system with 20 customers and 25 service facilities) the computation time increases proportionally for each iteration. Therefore, it is desirable to minimize the number of iterations required to locate X_o. Notations used in this chapter are listed below:

F.R denotes f(R).

INT(R) denotes the integer part of R.

RND denotes the next random number between 0 and 1(uniform).

VSM(i) denotes $V_i(M)*S_i(M)$.

VSU(i) denotes $V_i(U)*S_i(U)$.

X.EST denotes the estimate of X_o.

XM denotes $X_o(M)$.

V.1.1 Algorithm Analysis

The algorithms studied to minimize the number of iterations required to locate X_o are delineated below:

I) Bounded Binary Search (BBS) algorithm: As shown in Algorithm V.2, this algorithm keeps track of the upper and lower bounds of X_o during the iterations, and takes the average of the two bounds as the estimate of X_o for the next iteration. Note that the upper and lower bounds are updated simultaneously if $(LB)^i \le (XM)^{i+1} \le (UB)^i$. In a regular binary search algorithm, either the upper or lower bound is updated at an iteration. The justification for this simultaneous updates is given in Lemma V.1.II.

II) Bounded Interpolation (BI) algorithm: As shown in Algorithm V.3, this algorithm also keeps of the upper and lower bound of X_o, but applies interpolation to estimate X_o for the next iteration. As opposed to the BBS algorithm, only one bound (either the upper or lower) is updated at an iteration. On the other hand, the BI algorithm keeps track of f(UPPER.BOUND) and f(LOWER.BOUND) where "f" refers to the convolution algorithm, as shown in Algorithm V.1. Moreover, the BI algorithm also keeps track of X.EST and XM from the last iteration, which are denoted as LAST.X.EST and LAST.XM. LAST.X.EST and LAST.XM are used to interpolate the new X.EST. It is likely that either X.EST > LAST.X.EST or X.EST < LAST.X.EST. It would be easy, using analytical geometry, to show that the same formula can be used to evaluate DELTA, as shown in Algorithm V.3.

```
REM ======== [BOUNDED BINARY SEARCH] ALGORITHM =========
UPPER.BOUND = INITIAL.UPPER.BOUND:
LOWER.BOUND = INITIAL.LOWER.BOUND
X.EST = (UPPER.BOUND + LOWER.BOUND)/2
CALL CONVOLUTION.ALGORITHM
NUMBER.OF.ITERATIONS = 1
WHILE ( ABS(XM - X.EST) / X.EST ) > RELATIVE.ERROR
|   IF
|      XM<LOWER.BOUND
|         THEN
|            UPPER.BOUND=X.EST
|         ELSE
|            IF
|               LOWER.BOUND<=XM AND XM <= UPPER.BOUND
|                  THEN
|                     IF
|                        XM<=X.EST
|                           THEN
|                              LOWER.BOUND=XM:
|                              UPPER.BOUND=X.EST
|                           ELSE
|                              LOWER.BOUND=X.EST:
|                              UPPER.BOUND=XM
|                  ELSE
|                     LOWER.BOUND=X.EST
|   X.EST =(LOWER.BOUND+UPPER.BOUND)/2
|   CALL CONVOLUTION.ALGORITHM
|   NUMBER.OF.ITERATIONS = NUMBER.OF.ITERATIONS + 1
WEND
```

Algorithm V.2: The BBS Algorithm

```
===== <BOUNDED INTERPOLATION> ALGORITHM WITHOUT ADJUSTMENT =====
UPPER.BOUND = INITIAL.UPPER.BOUND:
LOWER.BOUND = INITIAL.LOWER.BOUND
SLOPE = (F.LOWER.BOUND - F.UPPER.BOUND)/(UPPER.BOUND - LOWER.BOUND):
DELTA=(UPPER.BOUND-F.UPPER.BOUND)/(1+SLOPE):
X.EST = UPPER.BOUND - DELTA
CALL CONVOLUTION.ALGORITHM
NUMBER.OF.ITERATIONS = 1
WHILE ( ABS(XM - X.EST) / X.EST ) > RELATIVE.ERROR
|  IF
|     XM<LOWER.BOUND
|        THEN
|           LAST.X.EST=LOWER.BOUND:
|           LAST.XM=F.LOWER.BOUND:
|           UPPER.BOUND=X.EST:
|           F.UPPER.BOUND=XM
|        ELSE
|           IF
|              UPPER.BOUND<XM
|                 THEN
|                    LAST.X.EST=UPPER.BOUND:
|                    LAST.XM=F.UPPER.BOUND:
|                    LOWER.BOUND=X.EST:
|                    F.LOWER.BOUND=XM
|  IF
|     LOWER.BOUND<=XM AND XM <= UPPER.BOUND
|        THEN
|           IF
|              XM<=X.EST
|                 THEN
|                    LAST.X.EST=LOWER.BOUND:
|                    LAST.XM=F.LOWER.BOUND:
|                    UPPER.BOUND=X.EST:
|                    F.UPPER.BOUND=XM
|                 ELSE
|                    LAST.X.EST=UPPER.BOUND:
|                    LAST.XM=F.UPPER.BOUND:
|                    LOWER.BOUND=X.EST:
|                    F.LOWER.BOUND=XM
|  SLOPE=(LAST.XM-XM)/(X.EST-LAST.X.EST):
|  DELTA=(X.EST-XM)/(SLOPE+1)
|  X.EST = X.EST-DELTA
|  CALL CONVOLUTION.ALGORITHM
|  NUMBER.OF.ITERATIONS = NUMBER.OF.ITERATIONS + 1
WEND
```

Algorithm V.3: The BI/O Algorithm

The lemmas below prove the correctness of the choices of the upper and lower bounds used by the two algorithms, as discussed above.

Lemma V.1.I

Let $(UB)^i$ denote the upper bound at the i_{th} iteration, $(LB)^i$ denote the lower bound at the i_{th} iteration, $(X_o)^i$ denote the estimate of X_o at the i_{th} iteration, and $(XM)^{i+1}$ denote $(X_o(M))^{i+1}$ which equals to $f((X_o)^i)$, then one of the following conditions must exist for the BI and BBS algorithms:

I) $(XM)^{i+1} \le (LB)^i \le (X_o)^i \le (UB)^i$;

II) $(LB)^i \le (XM)^{i+1} \le (X_o)^i \le (UB)^i$;

III) $(LB)^i \le (X_o)^i \le (XM)^{i+1} \le (UB)^i$;

VI) $(LB)^i \le (X_o)^i \le (UB)^i \le (XM)^{i+1}$.

<Proof> The binary search and interpolation mechanisms guarantee that $(LB)^i \le (X_o)^i \le (UB)^i$. It follows that the four conditions are mutually exclusive and collectively exhaustive. Q.E.D.

Lemma V.1.II

Let $(UB)^i$ denote the upper bound at the i_{th} iteration, $(LB)^i$ denote the lower bound at the i_{th} iteration,

$(X_o)^i$ denote the estimate of X_o at the i_{th} iteration, and $(XM)^{i+1}$ denote $(X_o(M))^{i+1}$ which equals to $f((X_o)^i)$, then the upper and lower bounds are determined as follows for the four conditions of Lemma V.1.I.

I) $(LB)^{i+1} = (LB)^i \quad \wedge \quad (UB)^{i+1} = (X_o)^i$;

II) $(LB)^{i+1} = (XM)^{i+1} \quad \wedge \quad (UB)^{i+1} = (X_o)^i$;

III) $(LB)^{i+1} = (X_o)^i \quad \wedge \quad (UB)^{i+1} = (XM)^{i+1}$;

VI) $(LB)^{i+1} = (X_o)^i \quad \wedge \quad (UB)^{i+1} = (UB)^i$.

<Proof> The lemma is proven for condition I. Other conditions follow by the same token.

From condition I of Lemma V.1.I, $(XM)^{i+1} \le (LB)^i \le (X_o)^i \le (UB)^i$

From the CMD property, $(XM)^{i+1} \le (X_o)^\infty \le (X_o)^i$

and by definition, $(LB)^i \le (X_o)^\infty \le (UB)^i$

Therefore, $(LB)^{i+1} = (LB)^i \quad \wedge \quad (UB)^{i+1} = (X_o)^i$ Q.E.D.

Note that in the BI algorithm, it is possible that an estimate from an interpolation is out of bound. Specifically, the estimate maybe samller than the lower bound in condition II of

Lemma V.1.II, and greater than the upper bound in condition III of Lemma V.1.II. On the other hand, f(LOWER.BOUND) is unknown in condition II while f(UPPER.BOUND) is unknown in condition III. Therefore, even though both of the new upper and lower bounds are known for the $(i+1)_{th}$ iteration, only one bound can be updated in the cases of condition II and III. In other words, the information about a tighter bound is not exploited. Let the BI algorithm without exploiting this information be denoted as BI/O, which is shown in Algorithm V.3.

It was observed by the author that this information can be employed to adjust X.EST. In theory, the adjustment is equivalent to fully exploiting the bound information. Let the BI algorithm with adjustment be denoted as BI/A, as shown in Algorithm V.4. Note that the only difference between BI/O and BI/A is the adjustment which appears 4 lines above the bottom of Algorithm V.4. The efficiency of BBS, BI/O, and BI/A are discussed in the next section.

```
====== <BOUNDED INTERPOLATION> ALGORITHM WITH ADJUSTMENT ========
UPPER.BOUND = INITIAL.UPPER.BOUND:
LOWER.BOUND = INITIAL.LOWER.BOUND
SLOPE = (F.LOWER.BOUND - F.UPPER.BOUND)/(UPPER.BOUND - LOWER.BOUND):
DELTA=(UPPER.BOUND-F.UPPER.BOUND)/(1+SLOPE):
X.EST = UPPER.BOUND - DELTA
CALL CONVOLUTION.ALGORITHM
NUMBER.OF.ITERATIONS = 1
WHILE ( ABS(XM - X.EST) / X.EST ) > RELATIVE.ERROR
|   IF XM<LOWER.BOUND
|         THEN
|            LAST.X.EST=LOWER.BOUND:
|            LAST.XM=F.LOWER.BOUND:
|            UPPER.BOUND=X.EST:
|            F.UPPER.BOUND=XM
|         ELSE
|            IF UPPER.BOUND<XM
|                  THEN
|                     LAST.X.EST=UPPER.BOUND:
|                     LAST.XM=F.UPPER.BOUND:
|                     LOWER.BOUND=X.EST:
|                     F.LOWER.BOUND=XM
|   IF LOWER.BOUND<=XM AND XM <= UPPER.BOUND
|         THEN
|            IF XM<=X.EST
|                  THEN
|                     CONDITION=2:
|                     LAST.X.EST=LOWER.BOUND:
|                     LAST.XM=F.LOWER.BOUND:
|                     UPPER.BOUND=X.EST:
|                     F.UPPER.BOUND=XM
|                  ELSE
|                     LAST.X.EST=UPPER.BOUND:
|                     LAST.XM=F.UPPER.BOUND:
|                     LOWER.BOUND=X.EST:
|                     F.LOWER.BOUND=XM:
|                     CONDITION=3
|   SLOPE=(LAST.XM-XM)/(X.EST-LAST.X.EST):
|   DELTA=(X.EST-XM)/(SLOPE+1)
|   X.EST = X.EST-DELTA
|   IF CONDITION=2 AND X.EST<XM
|         THEN
|            X.EST=XM
|         ELSE
|            IF CONDITION=3 AND X.EST>XM
|                  THEN
|                     X.EST=XM
|   CALL CONVOLUTION.ALGORITHM
|   NUMBER.OF.ITERATIONS = NUMBER.OF.ITERATIONS + 1
WEND
```

Algorithm V.4: The BI/A Algorithm

V.1.2 Algorithm Efficiency

The efficiency of the regular binary search algorithm is the order of $LOG_2(R)$. In other words, it would take 10 iterations to search a variable in an interval R to achieve a relative error of .001, where the relative error is defined as follows: (CURRENT.ESTIMATE - LAST.ESTIMATE)/CURRENT.ESTIMATE. The BBS algorithm takes advantage of the bounds, as shown in Lemma V.1.II. Therefore, it is expected to perform better than the regular binary search algorithm. Suppose that an XM evaluated from the convolution algorithm may fall on any point between the upper and lower bound (i.e. uniformly distributed), then the expected efficiency of the BBS algorithm would be of $LOG_4(R)$. In other words, it would take 5 iterations on the average to achieve a relative error of .001. On the other hand, if the distribution is not uniform, then the expected efficiency would deviate from 5 iterations.

The BI/O algorithm has looser bounds than the BBS algorithm, but takes advantage of the fact that, at equilibrium, X.EST = XM. Therefore, it is not clear whether BI/O will outperform BBS or not.

The BI/A algorithm not only takes advantage of the bounds, but also considers the fact that, at equilibrium, X.EST = XM; therefore, it is expected to perform better than both of the BBS and BI/O algorithms.

V.1.3 Simulation Experiments

A simulation program was written to validate the BBS, BI/O, and BI/A algorithms. The efficiency of these algorithms for different cases, as elaborated in Chapter IV.2.2, were compared based on the simulation results and conclusions drawn. A complete listing of the simulation program is available in Appendix I.

The experiments were based on a uniformly distributed random number generator (29, 30). The workloads of networks with one to twenty customers and two to twenty service facilities were generated using the random number generator. The algorithm used to initialize and simulate an experiment is delineated in Algorithm V.5. The stability condition, as elaborated in Chapter IV.2.2, is tested to insure that a unique solution exists. The algorithm used to test the stability condition is delineated in Algorithm V.6. In Algorithm V.6, if the case type turns out to be I, II, or III, then a unique solution exists. In these cases, the BBS, BI/O, and BI/A algorithms are invoked to evaluate X_0.

```
REM =============  SIMULATE AN EXPERIMENT  =================

MAX.VSU=0:
NUMBER.OF.ITERATIONS=0:
LOWER.BOUND=0:
UPPER.BOUND=0
NUMBER.OF.FACILITIES = INT(RND*19) + 2
NUMBER.OF.CUSTOMERS = INT(RND*20) + 1
VSM.INDEX = 0
FOR M = 1 TO NUMBER.OF.FACILITIES
|  VSM(M) = INT(RND*6) * RND
|  VSU(M) = INT(RND*4) * RND
|  IF
|     VSM(M)>0
|        THEN
|           VSM.INDEX = 1
|  IF
|     VSU(M)> MAX.VSU
|        THEN
|           MAX.VSU = VSU(M):
|           MAX.VSU.INDEX = M
NEXT M
```

Algorithm V.5: Initialize and Simulate an Experiment
-------------- --------------------------------------

```
REM ===== TEST STABILITY CONDITION TO IDENTIFY THE CASE.TYPE =====
.sk
MAX.XM =1/MAX.VSU
X.EST =0
CALL CONVOLUTION.ALGORITHM
IF
   XM<MAX.XM
     THEN
        CASE.TYPE=1
     ELSE
         IF
           VSM(MAX.VSU.INDEX)>0
              THEN
                 CASE.TYPE=2
              ELSE
                 X.EST=MAX.XM:
                 GOSUB 4000:
                 IF
                    XM<=MAX.XM
                       THEN
                          CASE.TYPE=3
                       ELSE
                          CASE.TYPE=4

REM ========== END OF STABILITY CONDITION TEST ==================
```

Algorithm V.6: The Stability Condition Test

V.1.4 Simulation Results

10,000 simulation experiments were conducted. The BBS, BI/O, and BI/A algorithm were applied to each simulation experiment to determine the number of iterations required to achieve a relative error of .001. The 10,000 experiments were partitioned into five groups. The statistical results of the experiments are shown in Table V.1, V.2, V.3, and V.4.

As analyzed in Chapter V.1.2, the simulation results also indicate that the efficiency of the BI/A algorithm is much better than that of the BBS algorithm.

It is interesting to note that the BI/O algorithm performs identical to the BI/A algorithm. Clearly, it implies that the adjustment does not adjust at all. Specifically, the X.EST's were always between LOWER.BOUND and UPPER.BOUND in the case of condition II and III of Lemma V.1.II. However, the BI/a algorithm is better from the theoretical point of view because it guarantees the same bounds as the BBS algorithm.

It was argued that if the outcome of XM is uniformly distributed between the upper and lower bound, then the efficiency of the BBS algorithm will be of $LOG_4(R)$. The simulation results indicate that it is $LOG_{2.5}(R)$ instead; suggesting that the outcome of XM tend to be closer to the upper (or lower) bound than X.EST.

A cross examination of Table V.1, V.2, V.3, and V.4 indicates that 91% of the simulation experiments turned out to be case I, 7% turned out to be case II, and 1% turned out to be case III. The performance of the algorithms for different cases is plotted in Figure V.1. It is clear that the BI/A (or the BI/O) algorithm should be adopted to evaluate X_o for case I and the BBS algorithm adopted for case III. The BI/A algorithm was used to implement TAD since the majority of experiments were found to be case I.

GROUP:STATISTICS	BI/A	BI/O	BBS
I:NO.OF.ITERATIONS	4648	4648	15065
I:NO.OF.REPLICATES	2000	2000	2000
I:MEAN	2.324	2.324	7.533
I:S.D.	3.125	3.125	2.096
II:NO.OF.ITERATIONS	4737	4737	15141
II:NO.OF.REPLICATES	2000	2000	2000
II:MEAN	2.369	2.369	7.571
II:S.D.	3.094	3.094	2.046
III:NO.OF.ITERATIONS	4756	4756	15101
III:NO.OF.REPLICATES	2000	2000	2000
III:MEAN	2.378	2.378	7.551
III:S.D.	3.768	3.768	2.086
VI:NO.OF.ITERATIONS	4601	4601	15071
VI:NO.OF.REPLICATES	2000	2000	2000
VI:MEAN	2.30	2.30	7.536
VI:S.D.	2.447	2.447	2.059
V:NO.OF.ITERATIONS	4955	4955	15139
V:NO.OF.REPLICATES	2000	2000	2000
V:MEAN	2.478	2.478	7.570
V:S.D.	5.349	5.349	2.041
GRAND MEAN	2.370	2.370	7.552
GRAND S.D.	3.692	3.692	2.066

Table V.1: Results of BI/A, BI/O, and BBS: Overall

GROUP:STATISTICS	BI/A	BI/O	BBS
I:NO.OF.ITERATIONS	3590	3590	13880
I:NO.OF.REPLICATES	1814	1814	1814
I:MEAN	1.979	1.979	7.652
I:S.D.	1.026	1.026	1.803
II:NO.OF.ITERATIONS	3624	3624	13909
II:NO.OF.REPLICATES	1816	1816	1816
II:MEAN	1.996	1.996	7.659
II:S.D.	1.103	1.103	1.794
III:NO.OF.ITERATIONS	3644	3644	13997
III:NO.OF.REPLICATES	1824	1824	1824
III:MEAN	1.998	1.998	7.674
III:S.D.	1.191	1.191	1.821
VI:NO.OF.ITERATIONS	3611	3611	13816
VI:NO.OF.REPLICATES	1812	1812	1812
VI:MEAN	1.993	1.993	7.625
VI:S.D.	1.059	1.059	1.817
V:NO.OF.ITERATIONS	3621	3621	13848
V:NO.OF.REPLICATES	1814	1814	1814
V:MEAN	1.996	1.996	7.634
V:S.D.	1.118	1.118	1.798
GRAND MEAN	1.990	1.990	7.649
GRAND S.D.	1.1	1.1	1.807

Table V.2: Results of BI/A, BI/O, and BBS: Case I

GROUP:STATISTICS	BI/A	BI/O	BBS
I:NO.OF.ITERATIONS	709	709	938
I:NO.OF.REPLICATES	145	145	145
I:MEAN	4.890	4.890	6.469
I:S.D.	7.580	7.580	3.029
II:NO.OF.ITERATIONS	632	632	937
II:NO.OF.REPLICATES	142	142	142
II:MEAN	4.451	4.451	6.599
II:S.D.	5.611	5.611	2.843
III:NO.OF.ITERATIONS	575	575	848
III:NO.OF.REPLICATES	134	134	134
III:MEAN	4.291	4.291	6.328
III:S.D.	6.892	6.892	2.846
VI:NO.OF.ITERATIONS	701	701	948
VI:NO.OF.REPLICATES	149	149	149
VI:MEAN	4.705	4.705	6.362
VI:S.D.	6.605	6.605	2.929
V:NO.OF.ITERATIONS	656	656	942
V:NO.OF.REPLICATES	145	145	145
V:MEAN	4.524	4.524	6.497
V:S.D.	16.08	16.08	2.970
GRAND MEAN	4.578	4.578	6.452
GRAND S.D.	9.399	9.399	2.926

Table V.3: Results of BI/A, BI/O, and BBS: Case II

GROUP:STATISTICS	BI/A	BI/O	BBS
I:NO.OF.ITERATIONS	163	163	77
I:NO.OF.REPLICATES	17	17	17
I:MEAN	9.588	9.588	4.529
I:S.D.	18.06	18.06	1.821
II:NO.OF.ITERATIONS	220	220	96
II:NO.OF.REPLICATES	19	19	19
II:MEAN	11.58	11.58	5.053
II:S.D.	18.74	18.74	2.057
III:NO.OF.ITERATIONS	161	161	77
III:NO.OF.REPLICATES	15	15	15
III:MEAN	10.73	10.73	5.133
III:S.D.	29.47	29.47	1.589
VI:NO.OF.ITERATIONS	226	226	105
VI:NO.OF.REPLICATES	24	24	24
VI:MEAN	9.417	9.417	4.375
VI:S.D.	5.915	5.915	1.965
V:NO.OF.ITERATIONS	312	312	131
V:NO.OF.REPLICATES	24	24	24
V:MEAN	13	13	13
V:S.D.	20.55	20.55	2.415
GRAND MEAN	10.93	10.93	4.909
GRAND S.D.	19.13	19.13	2.028

Table V.4: Results of BI/A, BI/O, and BBS: Case III

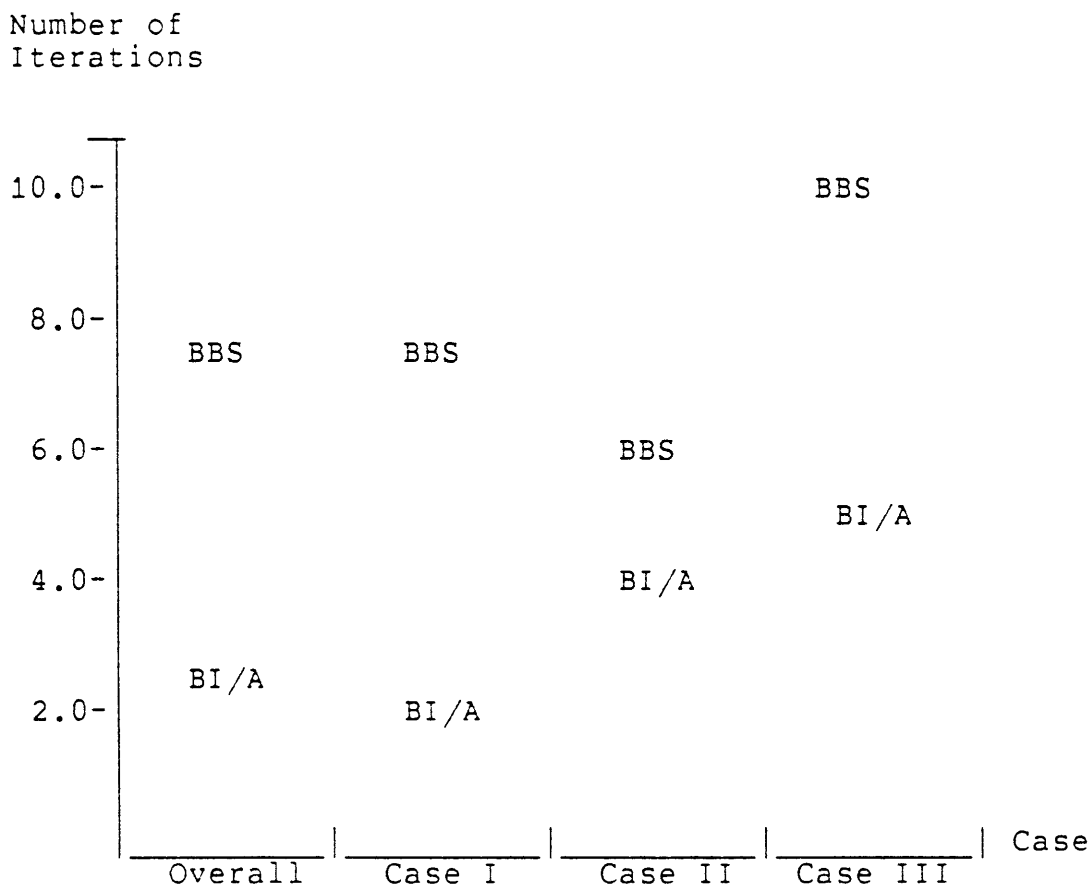


Figure V.1: Performance of BI/A vs. BBS

V.2 TAD

It is assumed that the reader has certain familiarity with the INFOPLEX data storage hierarchy (1, 46, 47, 55, 56, 93, 94). READ-THROUGH nad STORE-BEHIND are the two basic strategies employed in the data storage hierarchy. TAD was implemented based on these two strategies.

V.2.1 Significance of TAD

Contemporary analytic performance pacakages such as BEST/1 (9) and RESQ (77), though very powerful, cannot be applied to the INFOPLEX data storage hierarchy without modifications for the following (or some of the) reasons: a) they do not handle UAP; b) they do not handle generalized queueing networks; c) it takes a substantial effort to specify the routing definitions for any interesting data storage hierarchy model.

TAD has been designed to meet the above requirements. With TAD, one can not only capture the primary effect on performance due to UAP but also explore different design alternatives of the data storage hierarchy effectively with minimum effort in defining the model. It has been observed that: 1) it takes about 10 minutes to explore a design alternative using TAD in an interactive environment. On the other hand, it would take hours to obtain the desired information using simulation. 2) The cost is about five cents per design alternative using TAD; on the other hand, it would cost hundreds of dollars to explore the same design alternative using simulation.

V.2.2 Software Architecture of TAD

There are five major components in the TAD architecture:

I) A front end processor which interfaces with the INFOPLEX data storage hierarchy designer;

II) An error handler which handles validity checking and error recovery;

III) A model analyzer which computes the sum of products of visit-ratios and mean-service-times for each class of customers under different combinations of policies;

IV) A performance analyzer which computes performance measures;

V) A utility library which supports other components.

Component I supports the user with the following capabilities:

* Define a new model, save a defined model, and modify a saved model;
* Print out model parameters in a graphic form which depicts a data storage hierarchy model, as shown in Figure V.2;
* Select a combination of policies from a menu. The menu is shown in Figure V.3;

* Audit the sum of products of visit-ratios and mean-service-times for the selected combination of policies. A partial output of a P1L3 model is shown in Figure V.4, and the complete listing is available in Appendix II.
* Interface the performance measures of the selected combination of policies to plotting packages such as MINITAB.

Component II checks the validity of a new (or modified) model. Errors are reported interactively to the user for correction. For instance, the error handler checks whether mean-service-times are nonnegative; if the input is either a negative numeric variable or an alphanumerical variable, then an error recovery routine is invoked to inform the user of the mistake and take appropriate actions.

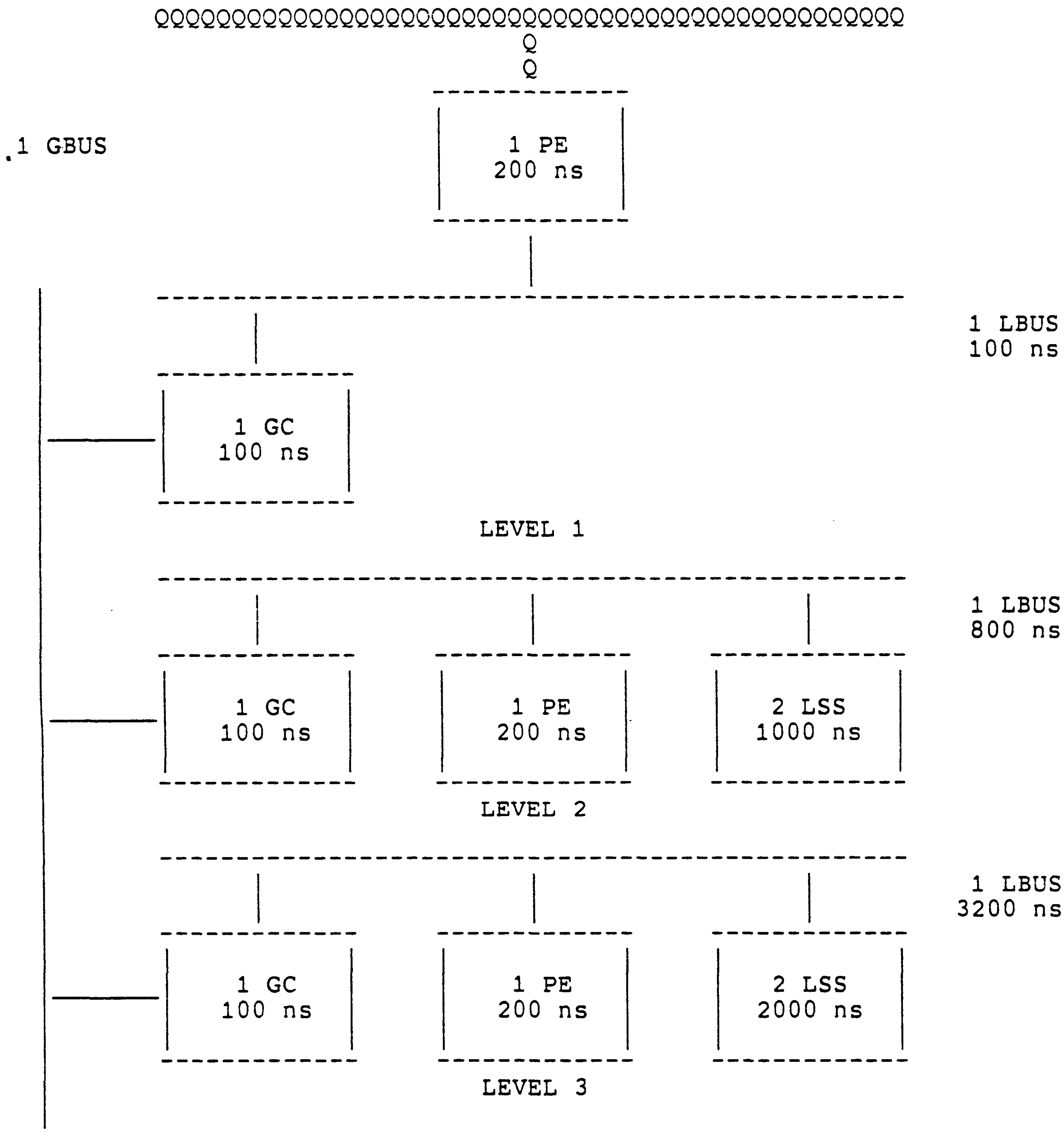


Figure V.2:
Figure V.2: Sample P1L3 Model Parameters in A Graphic Form.

```
*****************************************************************
YOU CAN SELECT THE COMBINATION OF POLICIES
BY ENTERING THE SUM OF THE POLICY NUMBERS BELOW:


10000 OPEN;          20000 CLOSED;

1000 PERCOLATE;      2000 PARALLEL;

100 RETRANSMIT;      200 RESERVE SPACE;

10 A (LOCALITY,READ%) POINT;    20 A LOCALITY SET GIVEN A READ%;
1 EQUAL PRIORITY;    2 STB LOW PRIORITY;


*****************************************************************
THE CURRENT COMBINATION OF POLICIES IS 21111 :
CLOSED, PERCOLATE, RETRANSMIT, A (LOCALITY,READ%) POINT,
AND EQUAL PRIORITY.
*****************************************************************


IS THIS WHAT YOU WANT? CONFIRM YES/NO:
YES
```

Figure V.3:
Menu with Different Combinations of Policies.

CHECK IN DSH LEVEL ONE PE.

NUMBER OF FACILITIES		LEVEL	VISIT-RATIO	SERVICE-TIME	VS-PRODUCT	CHAIN-TYPE
1	PE	1	.70000	200.000	140.0	1

READ-THROUGH-MESSAGE STOPS WHEN DATA IS FOUND;
IT'S FOLLOWED BY READ-THROUGH-RESULT-FOUND TRANSACTION.

READ-THROUGH-MSG.

NUMBER OF FACILITIES		LEVEL	VISIT-RATIO	SERVICE-TIME	VS-PRODUCT	CHAIN-TYPE
1	LBUS	1	.21000	100.000	21.0	1
1	GC	1	.21000	100.000	21.0	1
1	GBUS	1	.21000	100.000	21.0	1
1	GC	2	.21000	100.000	21.0	1
1	LBUS	2	.21000	100.000	21.0	1
1	PE	2	.21000	200.000	42.0	1
1	LBUS	2	.06300	100.000	6.3	1
1	GC	2	.06300	100.000	6.3	1
1	GBUS	2	.06300	100.000	6.3	1
1	GC	3	.06300	100.000	6.3	1
1	LBUS	3	.06300	100.000	6.3	1
1	PE	3	.06300	200.000	12.6	1

Figure V.4:
Sample Partial Audit Output of P1L3 Model.

Component III and component IV comprise the heart of TAD. Component III computes the sum of products of visit-ratios and mean-service-times for each class of customers of a model with any number of levels. The sum of products of visit-ratios and mean-service-times of each service facility plays a critical role in the solution of X_o. Theoretically, the determination of visit ratios involves nothing more than solving for a set of simultaneous linear equations. However, the coefficient matrix of the linear system explodes quickly for a generalized topology with a complex algorithm such as the READ-THROUGH and STORE-BEHIND data movement strategies. An angular structure matrix approach was developed (93) to calculate visit ratios for the INFOPLEX data storage hierarchy. The idea was to exploit the multi-class concept to model an algorithm. This idea was implemented in component III. This approach also simplifies the procedure to separate the unbalanced open chain flow from the main chain flow. As a result, performance measures such as utilizations are accurately estimated.

Since the sum of products of visit-ratios and mean-service-times are sensitive to different combinations of policies, different routines have to be invoked to perform the task. Currently, component III supports the following two policies: a) "open systems with a percolate down policy" and b) "closed systems with a percolate down policy." It would be easy to add new policies, such as "closed systems with a retransmit policy", simply by adding a subroutine to calculate the sum of products of visit-ratios and mean-service-times.

Component IV computes the following performance measures for open and closed systems: a) the overall system throughput and response time; b) facility utilization, mean queue length, and response time; and c) 99% probability buffer size. Note that: a) the overall system throughput and response time refer to the measures that the external world perceives; and b) the 99% buffer size refers to the buffer size that customers will find, with .99 probability, a buffer slot to queue in line for service at the facility.

V.2.3 Implementation of TAD

TAD was implemented on the PRIME 850 at the Sloan School of Management, M.I.T.. A complete listing of TAD is available in Appendix III. In addition to ease of use, it has been observed that use of TAD costs five cents per design design alternative, as depicted in Figure I.4. The validity of TAD was studied through the RESQ and GPSS simulation models, as presented in the next chapter.

CHAPTER VI

Validation Study Using INFOPLEX Data Storage Hierarchy Models

VI.1 VALIDATION OF PERFORMANCE MODELS

The development of a performance model involves characterizing the hardware and software components that comprise the system. For instance, the choices of the speeds of hardware devices, the use of replacement algorithms, and the service demands placed on facilities would change the characteristics, hence performance, of a model. A modeler may decide not to include certain features of the system structure (such as finite buffer length), and to represent other features (such as service demands), in a gross way. This will simplify the model in the belief that the abstraction will capture the primary effect on performance. In order to validate the predictive power of the model, it is ideal to compare the performance measures from the model with the measures from the actual system. However, it is usually unlikely to perform this kind of validity test in time, particularly because the system has not been built. After all, that is why the model was developed to begin with.

One way to validate a model is to compare it with other models with different level of details of a system. For instance, a detailed simulation model may be developed to compare its performance predictions with the predictions from an analytic model to test for consistencies. Any major discrepency between the simulated and analytic results would lead the designer to question the validity of the model. On the other hand, the validity of the model is not proven even if the simulation confirm the analytic results. Fortunately, the system designer's experiences over past systems can be applied to assess the validity of the model. Given the system has not been built, the combination of the system designer's experiences and the consistencies between the analytic and the simulation results is the most rigorous approach one can employ. The author has adopted this approach in this research. The validation of the analytic formulation is presented in this chapter through GPSS and RESQ simulation models using the INFOPLEX P5L4 and P1L3 models.

Three sets of notations were used in the INFOPLEX research to represent the components of data storage hierarchy models. They are listed in Table VI.1 for reference. These notations will be used interchangeably in the remainder of the thesis.

	GPSS	RESQ	TAD
Level 1 Device	D1	D1	PE
Level 1 Local Bus	LBUS1	L1	LBUS
Level 1 Gateway Controller	K1	K1	GC
Global Bus	GBUS	G	GBUS
Level 2 Gateway Controller	K2	K2	GC
Level 2 Local Bus	LBUS2	L2	LBUS
Level 2 Memory Request Processor	RRP2	M2	PE
Level 2 Local Storage Device 1	DRP21	D21	PE
Level 2 Local Storage Device 2	DRP22	D22	PE
Level 3 Gateway Controller	K3	K3	GC
Level 3 Local Bus	LBUS3	L3	LBUS
Level 3 Memory Request Processor	RRP3	M3	PE
Level 2 Local Storage Device 1	DRP21	D21	PE
Level 2 Local Storage Device 2	DRP22	D22	PE

Table VI.1:
Notations used by GPSS, RESQ, and TAD Programs

VI.2 THE P5L4 DATA STORAGE HIERARCHY MODEL

READ-THROUGH and STORE-BEHIND operations are the two basic strategies employed in the INFOPLEX data storage hierarchy. Lam79 (46, p.217-p.234) presented a detailed analysis of the P5L4 model using these two strategies. The structure of P5L4 is illustrated in Figure VI.1. The basic parameters used in the P5L4 model, which reflect the 1979 technology, are summarized in Figure VI.2.

```
                         REFERENCE SOURCE
                                Q
        QQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQ
              Q        Q        Q       Q        Q
              Q        Q        Q       Q        Q
           ------   ------   ------  ------   ------
GBUS       | PE |   | PE |   | PE |  | PE |   | PE |
           ------   ------   ------  ------   ------
              |        |        |       |        |
        -------------------------------------------------
              |                                             LBUS
        ------------          LEVEL 1
 _______|    GC    |
        ------------

        -------------------------------------------------
              |              |         |        |           LBUS
        -------------    --------  -------  -------
 _______|     GC    |    |  PE  |  | LSS |  | LSS |
        -------------    --------  -------  -------
                              LEVEL 2
        -------------------------------------------------
              |              |         |        |           LBUS
        -------------    --------  -------  -------
 _______|     GC    |    |  PE  |  | LSS |  | LSS |
        -------------    --------  -------  -------
                              LEVEL 3
        -------------------------------------------------
              |              |         |        |           LBUS
        -------------    --------  -------  -------
 _______|     GC    |    |  PE  |  | LSS |  | LSS |
        -------------    --------  -------  -------
                              LEVEL 4
```

KEY:
GBUS(GLOBAL BUS), LBUS(LOCAL BUS).
GC(GATEWAY CONTROLLER), PE(PROCESSOR ELEMENT)
LSS(LOCAL STORAGE SYSTEM)

Figure VI.1:
Structure of the P5L4
Data Storage Hierarchy Model.

DEGREE OF MULTIPROGRAMMING OF A CPU = 10.

SIZE OF DATA BUFFERS = 10.

READ/WRITE TIME OF A LEVEL 1 STORAGE DEVICE = 100 NANOSEC.

READ/WRITE TIME OF A LEVEL 2 STORAGE DEVICE = 1000 NANOSEC.

READ/WRITE TIME OF A LEVEL 3 STORAGE DEVICE = 10000 NANOSEC.

READ/WRITE TIME OF A LEVEL 4 STORAGE DEVICE = 100000 NANOSEC.

BUS SPEED = 10 MHZ.

BUS WIDTH = 8 BYTES.

SIZE OF A TRANSACTION WITHOUT DATA = 8 BYTES.

BLOCK SIZE AT LEVEL 1 = 8 BYTES.

BLOCK SIZE AT LEVEL 2 = 128 BYTES.

BLOCK SIZE AT LEVEL 3 = 1024 BYTES.

PERCENTAGE OF READ REQUESTS = 70%.

LOCALITY = 90%.

PROBABILITY OF OVERFLOW LEVEL 1 = 0.5

PROBABILITY OF OVERFLOW LEVEL 2 = 0.5

PROBABILITY OF OVERFLOW LEVEL 3 = 0.5

PROBABILITY OF OVERFLOW LEVEL 4 = 0

Figure VI.2:
Input Parameters of the P5L4
Data Storage Hierarchy Model.

VI.2.1 The P5L4 Simulation Model

The P5L4 simulation model of the INFOPLEX data storage hierarchy represents a basic structure from which extensions to include more processors and storage levels can be made. In the simulation model, there are five types of transactions supporting the READ-THROUGH and STORE-BEHIND operations. These transactions are: READ-THROUGH-REQUEST, READ-THROUGH-RESULT, OVERFLOW, STORE-BEHIND-REQUEST, and ACKNOWLEDGEMENT. Each type of transaction is handled differently. Furthermore, the same type of transaction is handled differently depending on whether the transaction is going into or out of a storage level. A detailed description of the simulation program is presented in Lam79 (42). The basic component of the P5L4 model is a facility and a number of data buffers, one for each type of transaction coming into the storage level and going out of the storage level. Three series of simulation studies have been conducted to predict the performance of the model with different parameters. The locality is always set to 90%.

The first series was conducted to obtain a well balanced system. The degree of parallelwasm in level 3 was increased by a factor of 5 from the basic parameter, and that of level 4 was increased by a factor of 10. Thwas was accomplwashed by decreasing the effective service times of the devices at these levels by 5 and 10 respectively. Finally, the model was run for three choices of block sizes: A(8,128,1024), B(8,64,512), and

C(8,64,256). The number 8 in choice A, for instance, means the block size transfer between level 1 and 2 is 8 bytes, and 128 means the transfer between level 2 and 3 is 128 bytes. This produces a fairly well-balanced system with the choice C(8,64,256).

The second series was based on the well balanced parameters. The model based on the 1979 technology with C(8,64,256) choice was run for 4 different request streams with different read percentages: .5, .7, .8, and .9.

The third series use 1985 technology assumptions. The bus speed was assumed to be 5 times faster than that used in the 1979 case. The level 1 storage device was assumed to be twice as fast in 1985 as in 1979. All other devices were estimated to be 10 times faster than 1979. Lastly, it was assumed that the directory search time would be reduced by one half in 1985. The model using 1985 technology assumption was run with the same 4 different request streams.

In sum, 10 simulation experiments were conducted to obtain performance measures. The results are used to compare with the abstract analytic model with corresponding parameters.

VI.2.2 The P5L4 Analytic Model

The P5L4 analytic model is highly abstracted from the simulation model. In order to analyze the INFOPLEX data storage hierarchy analytically, the following conditions have to be met:

1) A generalized topology has to be employed instead of the central server model;
2) Independent parallel tasks, such as broadcast and acknowledgement, should be allowed; and
3) A special structure to calculate the visit ratio should be developed.

TAD was developed to meet these conditions. The BI/A algorithm, as delineated in Chapter V.1, was implemented in TAD to compute the performance for a generalized INFOPLEX data storage hierarchy with any arbitrary number of global buses, local buses, gateway controllers, and local storage systems (1).

The parameters used in the 10 P5L4 simulation experiments were used in TAD to produce the corresponding performance measures. A detailed comparison is presented below.

VI.2.3 Comparison of the Results: Simulation vs Analytic Approach

Table VI.2 and Table VI.3 tabulate the system throughput and response time for the 10 studies. The comparison between the simulation and analytic results shows that the measures are highly consistent over these studies. The data indicate that the differences between the simulation and analytic results are within a factor of two.

It is argued, from the pattern of the measures, that if the simulation had been run long enough to eliminate the initial conditions, the measures would have converged to the analytic results. Another evidence that support this argument was a P1L3 model result where a deadlock occurred but the system throughput was 811 transactions per milli-second instead of zero for a simulation period of 1 milli-second (46).

SERIES.RUN	SIMULATED PERIOD	SIMULATION THROUGHPUT	ANALYTIC THROUGHPUT	SIM/ANA RATIO
1.79A	10 ms	176/ms	130/ms	1.36
1.79B	3 ms	458/ms	258/ms	1.78
1.79C	2 ms	721/ms	512/ms	1.4
2.79R50%	2 ms	450/ms	308/ms	1.46
2.79R70%	2 ms	721/ms	512/ms	1.41
2.79R80%	1 ms	1559/ms	767/ms	2.03
2.79R90%	1 ms	3239/ms	1531/ms	2.12
3.85R50%	.5 ms	2298/ms	1538/ms	1.49
3.85R70%	.3 ms	4320/ms	2561/ms	1.69
3.85R80%	.05 ms	15040/ms	3838/ms	3.92
3.85R90%	.05 ms	22760/ms	7656/ms	2.97

KEY:

1.79A: Series 1, 1979 Technology with the A choice.
3.85R90%: Series 3, 1985 Technology with 90% read.
ms: milli-second.

Table VI.2:
A Comparison of System Throughputs.

SERIES.RUN	SIMULATED PERIOD	SIMULATION RES. TIME	ANALYTIC RES. TIME	SIM/ANA RATIO
1.79A	10 ms	258580 ns	385956 ns	0.67
1.79B	3 ms	96260 ns	193733 ns	0.50
1.79C	2 ms	60940 ns	97620 ns	0.62
2.79R50%	2 ms	97580 ns	162586 ns	0.60
2.79R70%	2 ms	60940 ns	97621 ns	0.62
2.79R80%	1 ms	26790 ns	65138 ns	0.41
2.79R90%	1 ms	13440 ns	32655 ns	0.41
3.85R50%	.5 ms	19780 ns	32517 ns	0.60
3.85R70%	.3 ms	9940 ns	19524 ns	0.51
3.85R80%	.05 ms	2640 ns	13028 ns	0.21
3.85R90%	.05 ms	1760 ns	6531 ns	0.27

KEY:

1.79B: Series 1, 1979 Technology with the B choice.
2.79R50%: Series 2, 1979 Technology with 50% read.
ns: nono-second.

Table VI.3:
A Comparison of System Response Times.

A detailed analysis of the utilization patterns of the ten configurations also indicates that the simulation and the analytic results are highly consistent (58). Since the 1979 technology with choice A(8,128,1024) was simulated for the longest time(10 milli-seconds), its service facility utilizations are summarized in Table VI.4 to compare with those from TAD. The degree of consistency is convincing. The implication of the comparisons is clear: For the INFOPLEX data storage hierarchy architectural design, TAD is cost effective for exploring different design alternatives to compute the overall system performance and predict potential bottlenecks.

SERVICE FACILITY	SIMULATION UTILIZATION	ANALYTIC UTILIZATION
LEVEL 1 PE	.013	.009
GLOBAL BUS	.62	.588
LEVEL 1 LBUS	.02	.014
LEVEL 1 GC	.016	.014
LEVEL 2 LBUS	.10	.092
LEVEL 2 GC	.029	.026
LEVEL 2 PE	.028	.026
LEVEL 2 LSS	.03	.024
LEVEL 3 LBUS	.67	.64
LEVEL 3 GC	.02	.0197
LEVEL 3 PE	.016	.016
LEVEL 3 LSS	.043	.04
LEVEL 4 LBUS	1.0	1.0
LEVEL 4 GC	.007	.0077
LEVEL 4 PE	.007	.0077
LEVEL 4 LSS	.17	.195

Table VI.4:
Simulation vs TAD
Using 1979 Technology with Choice A.

VI.2.4 Implications of the P5L4 Validation Study

It was shown, in the last section, that the anlalytic formulation implemented in TAD was capable of producing performance measures which were consistent with the simulated results to within a factor of 2. Moreover, the utilization patterns were consistent between the analytic and simulated results to the second digit.

The predictive power of TAD was furthur demonstrated through a dramatic discovery. In a closer examination of the utilization patterns, Madnick (58) observed that the utilization of the level 3 local storage system obtained from simulation was significantly different from that from TAD. Furthur comparisons revealed that the difference was consistent across configurations. The puzzle gave rise to doubt about the validity of TAD.

Both the theory and implementation of TAD were scrutinized; however, no flaws were found. Consequently, the focus was shifted to the simulation. From the detailed simulation outputs (hundreds of pages), it was discovered that, of the two "level 3 local storage systems", one had a utilization which was different from that computed from TAD by a factor of 6, but the other one had a comparable utilization as that of TAD. The pattern was consistent across the configurations. Figure VI.3 illistrates one of the configurations: the average

FACILITY	UTILIZATION
GBUS	.616
LBUS1	.016
LBUS2	.102
LBUS3	.674
LBUS4	.995
DRP11	.014
DRP12	.013
DRP13	.015
DRP14	.013
DRP15	.013
KRP1	.016
KRP2	.029
RRP2	.028
KRP3	.020
RRP3	.016
KRP4	.007
RRP4	.007
DRP21	.028
DRP22	.030
DRP31	.280
DRP32	.043
DRP41	.174
DRP42	.206

<<<<<<<<<<<<<<<<<<
Deviate by 600% <
<<<<<<<<<<<<<<<<<<

Figure VI.3:
<u>Utilization Pattern of GPSS Program of P5L4 Model</u>

utilization of DRP31 (Level 3, Local Storage System 2) is .280 but the average utilization of DRP32 (level 3 local storage system 2) is .043 which is close to .04 as computed by TAD. The difference between .280 and .043 was too significant to be explained by sampling error. It became suspicious that the mistake may be on the simulation.

The simulation program (28 pages in length) was traced to uncover the puzzle. A typo was found on page 24 where a variable "DEX" was mistyped as "BEX". Figure VI.4 depicts the mistake. The puzzle was then solved because "BEX" has a different interpretation from "DEX" in the simulation program. Specifically, "BEX" assumed the value of bus service time while "DEX" assumed the value of local storage system service time. The typo was corrected and the utilizations were recalculated using the detailed simulation outputs. The corrected utilizations turned out to be consistent with those from TAD for all the configurations simulated.

This discovery helped establishing the reliability of TAD. On the other hand, the validity of the simulation results was further questioned. Two issues needed to be settled for simulation:

a) The simulation program has to be verified thoroughly; and
b) The simulation results has to be obtained in the steady-state.

```
FILE: GPSS54    VS1JOB    D24                    CONVERSATIONAL MONITOR SYSTEM

       ENTER       AOK2
       TRANSFER    ,ACK21

*-------------------------------------------------------------------*
*                                                                   *
* STORE-BEHIND TO L(3)                                              *
*                                                                   *
*-------------------------------------------------------------------*

STB23 ASSIGN       11,0

USE    MACRO       KRP2,XSKEX

SEND   MACRO       SOK2,SIK3,GBUS,XSBEX2,BVSKKS23

USE    MACRO       KRP3,XSKEX

SEND   MACRO       SIK3,SIR3,LBUS3,XSBEX2,BVSKRS3

USE    MACRO       RRP3,XSREX

       TRANSFER    .5,SWS31,SWS32

*************************
*                       *
* SB WRITE INTO D31     *
*                       *
*************************

SWS31 ASSIGN       11,0

SEND   MACRO       SIR3,SID31,LBUS3,XSBEX2,BVSRDS31

USE    MACRO       DRP31,XSBEX3

SEND   MACRO       SID31,SOK3,LBUS3,XSBEX3,BVSDKS3

       SPLIT       1,STB34
       ENTER       AOK3
       TRANSFER    ,ACK32

*************************
*                       *
* SB WRITE INTO D32     *
*                       *
*************************

SWS32 ASSIGN       11,0

SEND   MACRO       SIR3,SID32,LBUS3,XSBEX2,BVSRDS32

USE    MACRO       DRP32,XSDEX3

SEND   MACRO       SID32,SOK3,LBUS3,XSBEX3,BVSDKS3
```

Figure VI.4: The Typo in the Simulation Program for P5L4.

In order to fulfill these two requirements, a new simulation program was constructed using RESQ for the P1L3 model. The new RESQ simulation program follows Lam's (46) simulation program closely. The RESQ model, program, and results are presented in the next section.

VI.3 THE P1L3 DATA STORAGE HIERARCHY MODEL

The architecture of the P1L3 model is shown in Figure VI.5. Parameters for the P1L3 model was chosen to reflect 1979 processor and storage technology. The P5L4 model with balanced configuration was adapted to the P1L3 model by reducing the number of levels from 4 to 3 and the number of processors at level one from 5 to 1. Two key parameters that characterize the references are the locality level and the proportion of read and write requests in the reference stream.

A request to read a data item is handled by a data cache which has a directory service time REX. It is retrieved at a read service time DEX1 and sent back to the reference source. This probability is characterized by locality P. If the data item is not in the data cache, the request is passed down to lower storage levels, one by one. Therefore, there is a (1-P) probability that the read operation is passed down to LBUS1 which has a message transfer time BEXM. If the data item is found in the next lower level, it is returned through K1 back to D1 and returned to the reference source; otherwise, request is passed down to the next lower storage level. This is the basis for the mapping of the P1L3 read operation and workloads into a queueing netowrk model.

```
                         REFERENCE SOURCE
                                Q
          QQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQ
                                Q
                                Q
                              ------
  G                           | D1 |
                              ------
                                |
          ---------------------------------------------
              |                                      L1
          --------            LEVEL 1
 _________| K1   |
          --------


          ---------------------------------------------
              |          |          |         |      L2
          --------   --------   -------   -------
 _________| K2   |   | M2   |   | D21 |   | D22 |
          --------   --------   -------   -------
                              LEVEL 2

          ---------------------------------------------
              |          |          |         |      L3
          --------   --------   -------   -------
 _________| K3   |   | M3   |   | D31 |   | D32 |
          --------   --------   -------   -------
                              LEVEL 3
```

KEY:
G(GLOBAL BUS), L(LOCAL BUS).
K(GATEWAY CONTROLLER), M(MEMORY REQUEST PROCESSOR)
D(LOCAL STORAGE DEVICE)

Figure VI.5:
Architecture of the P1L3
Data Storage Hierarchy Model.

In a write operation, the addressed information is assumed to be updated in a data cache in zero time. After the data block is updated, an acknowledgement is returned to the reference source and the data block is sent to the next lower storage level through LBUS1, K1, GBUS, K2, LBUS2, MRP2, back to LBUS2, then to D21 or D22. Thus the effect of the update is propogated to lower storage levels.

VI.3.1 The P1L3 Simulation Model And Results

The RESQ simulation package was employed to conduct the simulation. A simulation program was developed to simulate the P1L3 model. The complete listing of the simulation program is available in Appendix IV. The input parameters used by the P1L3 model are summarized in Figure VI.6. A locality of .7 was assumed across the levels. A proportion of 70% of the arriving requests were assumed to be read requests.

The new RESQ program was verified thoroughly, partly due to the following factors:

I) RESQ allows the user to specify queue definitions and routing definitions independently, making the verification process easier; and

II) The variables used in the RESQ program were mnemonic, making the program easy to understand.

DEGREE OF MULTIPROGRAMMING OF A CPU = 20.

READ/WRITE TIME OF A LEVEL 1 STORAGE DEVICE = 100 NANOSEC.

READ/WRITE TIME OF A LEVEL 2 STORAGE DEVICE = 1000 NANOSEC.

READ/WRITE TIME OF A LEVEL 3 STORAGE DEVICE = 10000 NANOSEC.

BUS SPEED = 10 MHZ.

BUS WIDTH = 8 BYTES.

SIZE OF A TRANSACTION WITHOUT DATA = 8 BYTES.

BLOCK SIZE AT LEVEL 1 = 8 BYTES.

BLOCK SIZE AT LEVEL 2 = 64 BYTES.

BLOCK SIZE AT LEVEL 3 = 256 BYTES.

PERCENTAGE OF READ REQUESTS = 70%.

LOCALITY = 70%.

PROBABILITY OF OVERFLOW LEVEL 1 = 0.5

PROBABILITY OF OVERFLOW LEVEL 2 = 0.5

PROBABILITY OF OVERFLOW LEVEL 3 = 0.5

PROBABILITY OF OVERFLOW LEVEL 4 = 0

Figure VI.6:
Input Parameters of the P1L3
<u>Data Storage Hierarchy Model.</u>

The RESQ program was simulated for 200 CPU seconds. The key results are tabulated in Table VI.5. A key question is whether the simulation reached steady-state. This was concluded by the fact that the utilizations of D21 and D22, so does D31 and D32, were close to the second digits. The overall system throughput, perceived by the reference source, was 1.718 requests/micro-second. The overall system response time, perceived by the reference source was 11.56 micro-seconds. The complete listing of the RESQ simulation results is available in Appendix V.

VI.3.2 The P1L3 Analytic Model and Results

TAD was employed to conduct the analysis. The parameters used in TAD is the same as those of the RESQ simulation program, as shown in Figure VI.6. The overall system throughput, perceived by the reference source, was reported as 1.735 resuests/micro-second. The overall system response time perceived by the reference source was reported as 11.530 micro-seconds. The sums of products of visit-ratios and mean-service-times of each service facility was also reported by TAD. From these figures, the utilizations of all the facilities were computed directly from the formula $U_i = X_0 * (V_i * S_i)$. The resultant utilizations of all the facilities are also tabulated in Table VI.5 to compare with the RESQ simulation results. A complete listing of the TAD results is available in Appendix VI.

SERVICE FACILITY	SIMULATION UTILIZATION	TAD UTILIZATION	RELATIVE ERROR
G	.800	.808	.01
L	.245	.247	.0082
L2	.964	.973	.0093
L3	.985	.9994	.0146
D1	.615	.624	.0146
K1	.245	.247	.0082
K2	.363	.368	.0138
M2	.335	.339	.0119
K3	.129	.131	.0155
M3	.035	.137	.0148
D21(22)	.441	.442	.0023
D31(32)	.615	.629	.0228
Overall per	THROUGHPUT	THROUGHPUT	ERROR
micro-second	1.718	1.735	.0098
Overall in	RESP. TIME	RESP. TIME	ERROR
micro-second	11.56	11.53	.0026

Note: ERROR = |(SIMULATION-TAD)/SIMULATION)|

Table VI.5:
Comparative Results of P1L3 Model: Simulation vs. TAD.

VI.3.3 The Implications of the Comparative Results

The comparative results were tabulated in terms of absolute values and percentage difference between the RESQ simulation and TAD results, as shown in Table VI.5. The degree of consistency between TAD and the simulation results were striking: Both the overall system throughput and response time, perceived by the reference source, were accurate to within 1%. The utilizations of the service facilities were also consistent to the second decimal point. It is reasonable to conclude that TAD is a reliable tool for analyzing the INFOPLEX data storage hierarchy.

It is also important to recognize that at the architectural design stage, the significance of performance analysis is to abstract the essence of the system so that the overall system performance and potential bottlenecks can be identified. In this sense, the predictive power that TAD has demonstrated is more than satisfactory (32).

TAD was employed to explore new design alternatives. The results are presented in the next chapter.

CHAPTER VII

Technology Analysis and Design Alternative Explorations

It was shown, in Chapter VI, that TAD is a reliable and cost effective tool for exploring different design alternatives of the INFOPLEX data storage hierarchy. It would be interesting to apply TAD to analyze the performance of new design alternatives as a function of input parameters such as locality, read-percentage, and storage device speeds. This type of analysis would be expensive to conduct using simulation.

To be pragmatic, 1984 storage technologies were analyzed and the results were used to evaluate the performance of different data storage hierarchy models. Chapter VII.1 presents the results of the storage technology analysis. Chpater VII.2 presents a P1L4 configuration and a P1L5 configuration together with their corresponding analytic results produced from TAD.

VII.1 STORAGE TECHNOLOGY ANALYSIS

The following storage technologies were analyzed: ECL, MOS family, core, RAM-disk, Rigid-disk, Winchester-disk, optical-disk, and Mass Storage System. Price and performance data of these technologies were collected from 1) Auerbach Dataworld, 2) Computerworld Buyer's Guide, 3) Datapro70, 4) Data Sources, 5) Electronic Design, and manufacturers. Data from manufacturers, Datapro, and Computerworld were used to conduct the analysis while data from other sources were used to supplement the analysis. Specifically, data from Datapro were used to analyze the performance of 14-inch Winchester disk drives; data from Computerworld were used to analyze the performance of add-in memories; and data from IBM were used to analyze the performance of Mass Storage System. In addition, products were selected from all sources, whenever appropriate, to supplement the analysis.

Manufacturers' data are most reliable, but expensive to attain. The author has telephoned manufacturers, such as Storage Technology Corporation, for the current price and performance information. Moreover, the price and performance data of IBM hardware, as of June 1984, were collected. These data were used to validate data collected from other sources.

Dapapro has a comprehensive list of performance data about Winchester disk drives. The list includes more than 50 companies in addition to IBM. The performance data were analyzed statistically to assess the range of performance of 14-inch Winchester disks.

76 products were analyzed. The summary statistics, as shown in Table VII.1 and Figure VII.1, indicate that the means of the average seek time, average latency time, and average access time of 14-inch Winchester disk drives are 26.69 ms, 8.99 ms, and 38.68 ms respectively. It is interesting to observe that the minimum average seek time is 16 ms(by IBM 3380) which contributes to the majority of performance enhancement in the Winchester technology.

	AVERAGE SEEK TIME IN MILLI-SECOND(ms)	AVERAGE LATENCY TIME IN MILLI-SECOND(ms)	AVERAGE ACCESS TIME IN MILLI-SECOND(ms)
MEAN	29.69	8.99	38.68
MEDIAN	27.00	8.33	36.72
ST. DEV.	11.68	1.16	12.44
MINIMUM	16	8.3	24.3
MAXIMUM	65	12.5	77.5

Source: Datapro70.

Table VII.1:
Summary Statistics of 14-inch Winchester Disk Drives

AVERAGE SEEK TIME

MIDDLE OF INTERVAL	NUMBER OF OBSERVATIONS	
15	6	******
20	10	**********
25	23	***********************
30	22	**********************
35	1	*
40	2	**
45	6	******
50	2	**
55	4	****

AVERAGE LATENCY TIME
EACH * REPRESENTS 2 OBSERVATIONS

MIDDLE OF INTERVAL	NUMBER OF OBSERVATIONS	
8.5	52	**************************
9.0	1	*
9.5	6	***
10.0	12	******
10.5	5	***

AVERAGE ACCESS TIME

MIDDLE OF INTERVAL	NUMBER OF OBSERVATIONS	
25	9	*********
30	12	************
35	25	*************************
40	15	***************
45	1	*
50	2	**
55	6	******
60	6	******

Source: Datapro70.

Figure VII.1:
<u>Histograms of 14-inch Winchester Disk Drives</u>

Computerworld and Data Sources have comprehensive lists of storage technologies. They reflect the status-quo storage technologies in the open market. 110 products from Computerworld were used to analyze the MOS technology. 76 products used RAM devices while 34 products used DRAM devices. A t-test of the RAM group and the DRAM group indicated a 95% confidence interval of (-138, 47) in performance difference. In other words, the performance difference between RAM and DRAM is statistically insignificant. Therefore, they were lumped together as the MOS technology. The results are summarized in Table VII.2 and Figure VII.2. The mean and standard deviation of the MOS technology are 475 ns and 211 ns respectively. CMOS and NMOS were not included in the analysis because only a few products were available. Moreover, their price/performance characteristics were not significantly different from the MOS technology.

	MOS RAM IN NANO-SECOND(ns)	MOS DRAM IN NANO-SECOND(ns)	MOS RAM & DRAM IN NANO-SECOND(ns)
MEAN	461.41	506.74	475.42
MEDIAN	460	400	450
ST. DEV.	200.63	233.75	211.38
MINIMUM	58	150	58
MAXIMUM	1059	1200	1200

Source: Computerworld Buyer's Guide

Table VII.2:
Summary Statistics of MOS, MOS/RAM, and MOS/DRAM

MOS Technology (RAM & DRAM)

MIDDLE OF INTERVAL	NUMBER OF OBSERVATIONS	
100	4	****
200	8	********
300	14	**************
400	26	**************************
500	31	*******************************
600	5	*****
700	11	***********
800	5	*****
900	6	******

MOS/RAM

MIDDLE OF INTERVAL	NUMBER OF OBSERVATIONS	
100	4	****
200	7	*******
300	10	**********
400	12	************
500	24	************************
600	5	*****
700	8	********
800	3	***
900	3	***

MOS/DRAM

MIDDLE OF INTERVAL	NUMBER OF OBSERVATIONS	
200	1	*
300	4	****
400	14	**************
500	7	*******
600	0	
700	3	***
800	2	**
900	3	***

Source: Computerworld Buyer's Guide.

Figure VII.2:
Histograms of MOS, MOS/RAM, and MOS/DRAM

Several other storage technologies have different price/performance characteristics from MOS and Winchester technologies. However, only a few companies manufacture products with these technologies. They are ECL, RAM disk, and IBM 3850 Mass Storage System. Their average access times are .00005 ms, .3 ms, and 1000 ms respectively. Optical disks were reported (Electronic Design) to have an average access time of 450 ms and a price of .0007 cents/byte. It appeared that the optical disk technology fits between the Winchester technology and the IBM 3850 Mass Storage System. Unfortunately, the current optical disk technology produces write-once optical disks only. Therefore, unless a data storage hierarchy is designed for read-only applications, the optical disk technology is not usable. It was also observed that the core and rigid-disk technologies are incompetitive to other technologies. Therefore, the core, rigid-disk, and optical-disk technologies were eliminated from further analysis. In sum, 5 levels of storage technologies were identified. The results, as illustrated in Table VII.3 and Table VII.4, were used to configurate new data storage hierarchy models and conduct performance analyses.

LEVEL	TECHNOLOGY	AVERAGE ACCESS TIME IN MILLI-SECOND	UNIT PRICE IN DOLLAR	UNIT CAPACITY IN MEGABYTE	¢/BYTE
1	ECL	.00005	175,000	1	17.5
2	MOS	.00065	2,800	1	.28
3	RAM-DISK	.3	120,000	48	.25
4	WINCHESTER	24.3	86,310	2,500	.0034
5	IBM 3850	1000	236,000	236,000	.00028

LEVEL	EXAMPLE PRODUCT	SOURCE	DATE
1	DENELCOR INC.	COMPUTERWORLD	4/84
2	TREND/STANDARD MEMORIES INC.	COMPUTERWORLD DATA SOURCES	4/84 7/84
3	STC 4305, SERIES 6	DATAPRO70 STC	8/83 7/84
4	IBM/3380/A04	IBM	4/84
5	IBM/3851/A31	IBM	6/84

Table VII.3:
Data Storage Hierarchy using 1984 Technologies

VII.2 DESIGN ALTERNATIVE EXPLORATIONS

VII.2.1 P1L4 Configuration

A P1L4 configuration, as shown in Table VII.4, was proposed based on the results summarized in Table VII.3. To be conservative, the average access time of level 1 was doubled to 100 nano-seconds. It was also assumed that the system is closed with a population of 50 customers and a probability of .5 to overflow between levels. The "percolate, zero retransmit rate, and equal priority strategy" was used. The configuration would have a total storage capacity of 13 gigabytes at an expense of $.9 million for storage devices.

The P1L4 model is summarized in Figure VII.3. The analytic results, as a function of read-percentage and locality, are tabulated in Table VII.5 and plotted in Figure VII.4 and Figure VII.5. The analysis indicates that a throughput of 1.5 requests/micro-second and a response time of 33 micro-seconds would be achieved at a locality of .95 for a read-only data storage hierarchy. The performance would deteriorate as locality and read-percentage decrease.

LEVEL	UNIT PRICE IN DOLLAR	UNIT CAPACITY IN MEGABYTE	NUMBER OF UNITS	TOTAL CAPACITY IN MEGABYTE	TOTAL PRICE IN DOLLAR
1	175,000	1	1	1	175,000
2	2,800	1	8	8	22,400
3	120,000	48	2	96	240,000
4	86,310	2,529	5	12,600	431,550
			TOTAL	12,705	868,950

Table VII.4:
P1L4 Configuration using 1984 Technologies

```
                QQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQ
                                        Q
                                        Q
                                  -------------
  1 GBUS                          |   1 PE    |
                                  |  100 ns   |
                                  -------------
                                        |
   |            ------------------------------------------------------
   |                  |                                                 1 LBUS
   |                  |                                                 20 ns
   |            -------------
   |    --------|   1 GC    |
   |            |  100 ns   |
   |            -------------
   |                                  LEVEL 1
   |            ------------------------------------------------------
   |                  |                 |                 |             1 LBUS
   |                  |                 |                 |             160 ns
   |            -------------     -------------     -------------
   |    --------|   1 GC    |     |   1 PE    |     |   8 LSS   |
   |            |  100 ns   |     |  100 ns   |     |  650 ns   |
   |            -------------     -------------     -------------
   |                                  LEVEL 2
   |            ------------------------------------------------------
   |                  |                 |                 |             1 LBUS
   |                  |                 |                 |             640 ns
   |            -------------     -------------     -------------
   |    --------|   1 GC    |     |   1 PE    |     |   2 LSS   |
   |            |  100 ns   |     |  100 ns   |     | 300000 ns |
   |            -------------     -------------     -------------
   |                                  LEVEL 3
   |            ------------------------------------------------------
   |                  |                 |                 |             1 LBUS
   |                  |                 |                 |            2560 ns
   |            -------------     -------------     -------------
   |    --------|   1 GC    |     |   1 PE    |     |   5 LSS   |
   |            |  100 ns   |     |  100 ns   |     |24300000 ns|
   |            -------------     -------------     -------------
   |                                  LEVEL 4
   |
```

Figure VII.3:
P1L4 Configuration Using 1984 Technologies

READ%	LOCALITY	RESPONSE TIME(RT)	THROUGHPUT (TP)	ln(RT)	ln(TP)
0.95	0.10	193904640	0.0000003	19.0829	-15.1709
0.95	0.20	139803520	0.0000004	18.7557	-14.8437
0.95	0.30	97669040	0.0000005	18.3971	-14.4851
0.95	0.40	66005400	0.0000008	18.0052	-14.0932
0.95	0.50	43316824	0.0000012	17.5840	-13.6720
0.95	0.60	28107620	0.0000018	17.1515	-13.2395
0.95	0.70	18882168	0.0000026	16.7537	-12.8417
0.95	0.80	14144648	0.0000035	16.4648	-12.5528
0.95	0.90	12399322	0.0000040	16.3331	-12.4211
0.95	0.95	12181164	0.0000041	16.3154	-12.4034
0.97	0.10	192871104	0.0000003	19.0775	-15.1655
0.97	0.20	137631072	0.0000004	18.7401	-14.8281
0.97	0.30	94609616	0.0000005	18.3653	-14.4532
0.97	0.40	62279432	0.0000008	17.9471	-14.0351
0.97	0.50	39113256	0.0000013	17.4820	-13.5699
0.97	0.60	23583856	0.0000021	16.9761	-13.0640
0.97	0.70	14164042	0.0000035	16.4662	-12.5542
0.97	0.80	9326712	0.0000054	16.0484	-12.1364
0.97	0.90	7544574	0.0000066	15.8363	-11.9243
0.97	0.95	7321821	0.0000068	15.8064	-11.8943
0.99	0.10	191837568	0.0000003	19.0722	-15.1601
0.99	0.20	135458624	0.0000004	18.7242	-14.8122
0.99	0.30	91550208	0.0000005	18.3324	-14.4204
0.99	0.40	58553504	0.0000009	17.8854	-13.9734
0.99	0.50	34909776	0.0000014	17.3683	-13.4563
0.99	0.60	19060260	0.0000026	16.7631	-12.8511
0.99	0.70	9446190	0.0000053	16.0611	-12.1491
0.99	0.80	4508903	0.0000111	15.3216	-11.4095
0.99	0.90	2689838	0.0000186	14.8050	-10.8930
0.99	0.95	2462477	0.0000203	14.7167	-10.8047
1.00	0.10	191320800	0.0000003	19.0695	-15.1574
1.00	0.20	134372416	0.0000004	18.7161	-14.8041
1.00	0.30	90020496	0.0000006	18.3155	-14.4035
1.00	0.40	56690560	0.0000009	17.8531	-13.9411
1.00	0.50	32808080	0.0000015	17.3062	-13.3942
1.00	0.60	16798564	0.0000030	16.6368	-12.7248
1.00	0.70	7087510	0.0000071	15.7738	-11.8618
1.00	0.80	2100407	0.0000238	14.5576	-10.6456
1.00	0.90	262757	0.0001903	12.4790	-8.5670
1.00	0.95	32969	0.0015166	10.4033	-6.4913

Table VII.5:
P1L4 Analytic Results

```
ln(RESPONSE TIME)
 19.1+      4
     -           4    4
     -                     4
     -                          4    D
     -                               3    D
 16.2+                                    2    2    D  D
     -                                    A         C  C
     -                                         B    B
     -                                         A       B
     -
 13.3+
     -                                              A
     -
     -
     -
 10.4+                                                 A
       +---------+---------+---------+---------+---------+locality
     0.00      0.20      0.40      0.60      0.80      1.00
```

KEY READ%: 100% (A); 99% (B); 97% (C); 95% (D).

```
ln(THROUGHPUT)
 -6.4+                                                 A
     -
     -
     -
     -                                              A
 -9.3+
     -
     -                                         A
     -                                              B  B
     -                                    A    B    C  C
-12.3+                                    2    2    D  D
     -                               3    D
     -                          4    D
     -                     4
     -           4    4
-15.2+      4
       +---------+---------+---------+---------+---------+locality
     0.00      0.20      0.40      0.60      0.80      1.00
```

Figure VII.4:
<u>P1L4 Analytic Results</u>

VII.2.2 P1L5 Configuration

A P1L5 configuration, as shown in Table VII.6, was also proposed. It uses exactly the same assumptions as the P1L4 configuration, as described in Chapter VII.2.1. In addition, the IBM 3850 Mass Storage System was proposed as the fifth level of the storage hierarchy. The configuration would have a total storage capacity of 1200 gigabytes at an expense of $3.8 million for storage devices.

The analytic results, as a function of read-percentage and locality, are tabulated in Table VII.7 and plotted in Figure VII.5. The analysis indicates that STB operations has a significant impact over the system performance when the average access time at the bottom level is relative slow and the degree of parallelism is low. This observation suggests that a "coalescence" strategy would be useful to enhance performance.

LEVEL	UNIT PRICE IN DOLLAR	UNIT CAPACITY IN MEGABYTE	NUMBER OF UNITS	TOTAL CAPACITY IN MEGABYTE	TOTAL PRICE IN DOLLAR
1	175,000	1	1	1	175,000
2	2,800	1	8	8	22,400
3	120,000	48	2	96	240,000
4	86,310	2,520	1	2,520	86,310
5	664,000	236,000	5	1180000	3320000
			TOTAL	1,182,625	3,843,710

Table VII.6:
P1L5 Configuration using 1984 Technologies

READ%	LOCALITY	RESPONSE TIME(RT)	THROUGHPUT (TP)	ln(RT)	ln(TP)
0.95	0.10	7231722496	0.0000000	22.7017	-18.7897
0.95	0.20	4702683136	0.0000000	22.2714	-18.3594
0.95	0.30	2963604992	0.0000000	21.8097	-17.8976
0.95	0.40	1829829888	0.0000000	21.3275	-17.4155
0.95	0.50	1141326080	0.0000000	20.8554	-16.9434
0.95	0.60	762689024	0.0000001	20.4524	-16.5403
0.95	0.70	583111552	0.0000001	20.1839	-16.2719
0.95	0.80	516416384	0.0000001	20.0624	-16.1504
0.95	0.90	501025984	0.0000001	20.0322	-16.1201
0.95	0.95	500064064	0.0000001	20.0302	-16.1182
0.97	0.10	7173446656	0.0000000	22.6936	-18.7816
0.97	0.20	4591169536	0.0000000	22.2474	-18.3354
0.97	0.30	2815484416	0.0000000	21.7584	-17.8464
0.97	0.40	1657845248	0.0000000	21.2288	-17.3168
0.97	0.50	954846208	0.0000001	20.6771	-16.7650
0.97	0.60	568226944	0.0000001	20.1580	-16.2460
0.97	0.70	384865344	0.0000001	19.7684	-15.8564
0.97	0.80	316762304	0.0000002	19.5737	-15.6616
0.97	0.90	301047552	0.0000002	19.5228	-15.6108
0.97	0.95	300065472	0.0000002	19.5195	-15.6075
0.99	0.10	7115171840	0.0000000	22.6855	-18.7735
0.99	0.20	4479656960	0.0000000	22.2228	-18.3108
0.99	0.30	2667366400	0.0000000	21.7043	-17.7923
0.99	0.40	1485865728	0.0000000	21.1193	-17.2072
0.99	0.50	768378240	0.0000001	20.4598	-16.5478
0.99	0.60	373784000	0.0000001	19.7392	-15.8272
0.99	0.70	186626944	0.0000003	19.0446	-15.1326
0.99	0.80	117109344	0.0000004	18.5786	-14.6666
0.99	0.90	101069216	0.0000005	18.4313	-14.5193
0.99	0.95	100066816	0.0000005	18.4213	-14.5093
1.00	0.10	7086033920	0.0000000	22.6814	-18.7694
1.00	0.20	4423901184	0.0000000	22.2103	-18.2983
1.00	0.30	2593308160	0.0000000	21.6762	-17.7642
1.00	0.40	1399878400	0.0000000	21.0596	-17.1476
1.00	0.50	675151104	0.0000001	20.3304	-16.4184
1.00	0.60	276580672	0.0000002	19.4380	-15.5260
1.00	0.70	87537536	0.0000006	18.2876	-14.3756
1.00	0.80	17308332	0.0000029	16.6667	-12.7547
1.00	0.90	1218753	0.0000410	14.0133	-10.1013
1.00	0.95	151875	0.0003292	11.9308	-8.0188

Table VII.7:
P1L5 Analytic Results

```
ln(RESPONSE TIME)
             22.7+      4
                 -           4    4
     -                    4
     -                         4    D
     -                              2    2    2    2  2
 19.1+                              A    B
     -                                   A    B    B  B
     -
     -                                        A
     -
 15.5+
     -
     -                                             A
     -
     -
 11.9+                                                A
       +---------+---------+---------+---------+---------+locality
     0.00      0.20      0.40      0.60      0.80      1.00
```

KEY READ%: 100% (A); 99% (B); 97% (C); 95% (D).

```
ln(THROUGHPUT)
 -8.0+                                                A
     -
     -
     -                                             A
     -
-11.6+
     -
     -                                        A
     -
     -                                   A    B    B  B
-15.2+                              A    B
     -                              2    2    2    2  2
     -                         4    D
     -                    4
     -          4    4
-18.8+     4
       +---------+---------+---------+---------+---------+locality
     0.00      0.20      0.40      0.60      0.80      1.00
```

Figure VII.5:
<u>P1L5 Analytic Results</u>

VII.3 DISCUSSION

The analysis conducted in this chapter has demonstrated the power of TAD in providing insights into the behavior of the INFOPLEX data storage hierarchy. The cost-effectiveness of TAD also makes it attractive for the designer to explore different configurations with different storage capacities and expenses. Future research should be focused on the enhancement of distributed control algorithms based on analyses conducted through TAD, and on the enhancement of TAD itself.

CHAPTER VIII

Summary and Future Directions

A research built upon past researh efforts has been conducted. As a result of the integral effort, a technique has been developed to compute performance measures for distributed systems with unbalanced flows due to asynchronously spawned parallel tasks. With this technique, a cost effective architectural design tool, TAD, has been developed for the INFOPLEX data storage hierarchy models. Comparisons between the performance measures computed from TAD and those from detailed simulation studies indicate very high consistencies. It is clear that TAD is an attractive tool for exploring different INFOPLEX data storage hierarchy design alternatives.

VIII.1 SUMMARY OF THESIS

Chapter I of the thesis provided a rationale for a performance oriented software engineering methodology. Major accomplishments of this thesis were also listed.

The background and motivation of this research is the INFOPLEX database computer project. The motivation for using analytic product form queueing network models to analyze the INFOPLEX data storage hierarchy and the background queueing theory which is essential to the development of this research were presented in chapter II.

The existence problem of the product form solution for systems with unbalanced flows was discussed in chapter III. It has been concluded that the product form solution does not exist in general for systems with unbalanced flows.

An analytic formulation was presented in chapter IV to model and analyze distributed systems with unbalanced flows. A cutting technique was developed to approximate the performance of distributed systems with UAP. The stability conditions were identified. Priority schedualing of distributed systems with unbalanced flows was also addressed.

Chapter V extended the theory developed in chapter IV to study its applicability. The BI/A, BI/O and BBS algorithms were developed. These algorithms were studied, uisng simulation, to compare their efficiency. It was found that for the majority of networks (with 1 to 20 customers and 2 to 20 service facilities), the BI/A and BI/O algorithms took 1.79 iterations on the average to locate the equilibrium system throughput, X_o. The BBS algorithm took 7.65 iterations on the average to X_o All the algorithms outperform the conventional binary search algorithms which would take 10 iterations. The BI/A algorithm was implemented in a software package called TAD (Technique for Architectural Design) to evaluate the performance of different design alternatives of the INFOPLEX data storage hierarchy models.

Chapter VI presented the validiction study of TAD using INFOPLEX P5L4 ana P1L3 models. The study was conducted through the GPSS and RESQ simulation packages. Highly consistent results have been observed.

Chapter VII explored new disign alternatives using TAD. In addition to ease of use, it was observed that the use of TAD costs five cents per design alternative; whereas, it may not be possible to attain steady-state results of a single design alternative using simulation for $100. Better design alternatives were discovered and analyzed.

VIII.2 FUTURE DIRECTIONS

This thesis has provided an analytic framework for performance evaluation of the INFOPLEX data storage hierarchy models. With this foundation, future research can be conducted in the following directions:

I) More extensive validations of the analytic results both in terms of simulation and measurement of the actual system: Simulation studies with longer periods and with confidence interval estimates should be conducted before the actual system is built. RESQ(Sauer82) is a state-of-the-art tool that can be employed for future simulation studies.

II) The exploration of data storage hierarchy with more than four levels and with different data movement strategies:

The key advantage of the INFOPLEX data storage hierarchy is the extendability to any arbitrary number of levels. Different data movement strategies should be studied with an arbitrary number of levels to compare their performance. TAD is currently designed for an arbitrary number of levels with a percolate strategy. It can be employed to study the impact of the number of levels on data movement strategies. The extendability of TAD also offers an easy way to study different data movement strategies.

III) The extension of TAD to incorporate other features of the system, such as priority treatment, to obtain more accurate performance measures. Alternatively, the whole data storage hierarchy can be perceived as a composite service facility to be interfaced with the functional hierarchy. The closed system alternative makes the composition possible.

IV) Workload characterization of the INFOPLEX data storage hierarchy: The mean-service-times and visit-ratios play a critical role in the computation of performance measures of a model. New design decisions should be incorporated into the performance model to revise these parameters.

The development of TAD and the comparison of the TAD results with simulation results opens a door for a series of exciting researches. Future INFOPLEX research in the performance area should address the above issues.

BIBLIOGRAPHY

Abbreviations used in the references:

CACM Communication of the ACM
IEEE Institute of Electronic and Electrical Engineering
JACM Journal of the ACM

1. Abraham, M., "Data Storage Hierarchy Design," Thesis Proposal, MIT Sloan School, 1982.

2. Agrawal, S.C. and Buzen, P.J., "The Aggregate Server Method for Analyzing Serialization Delays in Computer Systems," BGS Systems, INc., CSD-TR-384, February 1982.

3. Allen, A.O., "Queueing Models of Computer Systems," IEEE Computers, April, 1980

4. Bard, Y., The Modeling of Some Scheduling Strategies for an Interactive Computer System, in Computer Performance, K.M. Chandy and M. Reiser (Eds.), Elsevier North-Holland, Inc., New York, 1977, pp. 113-138.

5. Baskett, F., Chandy, K.M., Muntz, R.R., and Palacios, J. "Open, Closed, and Mixed Networks with Different Classes of Customers," JACM 22,2(April 1975), 248-260.

6. Browne, J.C., Chandy, K.M., Hogarth, J., and Lee, C. "The Effect on Throughput in Multi-processing in a Multi-programming environment," IEEE Trans. Computers, Vol. 22, August 1973, pp 728-735.

7. Burke, P.J. "Output Process and Tandem Queues", Symposium on Computer-Communications Networks and Teletraffic Polytechnique Institute of Brooklyn, pp. 419-428, April 4-6, 1972

8. Buzen, J.P. "Queueing Network Models of Multiprogramming," Ph.D. Dissertation, Div. Eng. Appl. Sci., Harvard University, Cambridge, MA, 1971.

9. Buzen, J.P. "Computational Algorithms for Closed Queueing Networks with Exponential Servers," CACM 16,9(Sept. 1973), 527-531. Cambridge, MA, 1971.

10. Buzen, J.P., and Gagliardi, "The Evolution of Virtual Machine Architecture," AFIPS Conf. Proc. 42 (1973 NCC), pp. 291-301.

11. Buzen, J.P. "Cost Effective Analytic Tools for Computer Performance Evaluation, " COMPCON75.

12. Buzen, J.P. "Operational Analysis: The Key to the New Generation of Performance Prediction Tools," COMPCON76.

13. Buzen, J.P. "Fundamental Operational Laws of Computer System Performance, " Acta Informatica 7, 167-182 (1976).

14. Buzen, J.P. "Operational Analysis: An Alternative to Stochastic Modleing," Performance of Computer Installation, North-Halland Company, 1978.

15 Buzen, J.P., and Denning, P.J., "Operational Treatment of Queue Distributions and Mean Value Analysis," CSD-TR-309, Perdue University (August 1979).

16. Buzen, J.P. and Denning, P.J. "Measuring and Calculating Queue Length Distributions," IEEE, Computer, April, 1980, pp. 33-44.

17. Chandy, K.M., Herzog, U., and Woo, L.S., "Parametric Analysis of Queueing Networks," IBM J. of Research and Development 19, 1 pp. 43-49 (January 1975).

18. Chandy, K.M., Herzog, U., and Woo, L.S., "Approximate Analysis of General Queueing Networks," IBM J. of Research and Development 19, 1 pp. 50-57 (January 1975).

20. Chandy, K.M., Howard, J.H., and Towsley, D.F., "Product Form and Local Balance in Queueing Networks," JACM 24, 2 pp. 250-263 (April 1977)

21. Chandy, K.M., and Sauer, C.H., "Approximate Methods for Analysis of Queueing Network Models of Computer Systems," Computing Surveys 10, 3 pp. 263-280 (September 1978).

22. Chandy, K.M., and Sauer, C.H., "Computational Algorithms for Product Form Queueing Networks," RC-7590, IBM Research, Yorktown Heights, N.Y. (November 1979). CACM 23, 10 (October 1980)

23. Courtois, P.J., "Decomposability, Instabilities and Saturation in Multiprogramming Systems," Communications of the ACM 18, 7 pp. 368-371 (July 1975).

24. Courtois, P.J., Decomposability: Queueing and Computer System Applications, Academic Press, Inc., New York (1977).

25. Courtois, P.J., "Exact Aggregation in Queueing Networks," Proc. First Meeting AFCET-SMF, Paris (September 1978).

26. Cox, D.R. "A Use of Complex Probabilities in the Theory of Stochastic Processes," Proc. Cambridge Philos. Soc. 51, (1955), pp. 313-319.

27. Cox, D.R. and Miller, H.D., The Theory of Stochastic Processes, Wiley, New York, (1965).

28 Denning P.J. and Buzen,J.P. "The Operational Analysis of Queueing Network Models," ACM Computing Surveys, Vol.10, No. 3, Sept. 1978, pp. 225-261.

29. Drake, A.W., Fundamentals of Applied Probability Theory, McGraw- Hill, New York (1967).

30. Feller, W., An Introduction to Probability Theory and Its Implications, Wiley, New York (1968).

31. Ferrari, D., Computer System Performance Evaluation, Prentice-Hall, Englewood Cliffs, N.J. (1978).

32. Gagliardi, Ugo, Lectures on Software Engineering, Harvard University, 1982.

33. Geist, R.M. and Trivedi, K.S., "Optimal Design of Multilevel Storage Hierarchies," IEEE Transactions on Computers, Vol. c31, no.3, March 1982.

34. Goldberg, A. ,Popek, G., and Lavenberg, S. "A Validated Distributed System Performance Model," Performance'83, pp 251-268.

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

36. Goyal, A. and Agerwala, T. "Performance Analysis of Future Shared Storage Systems," IBM J. Res. Develop, Vol. 28, No.1, January, 1984

37. Graham, G.S. "Guest Editor's Overview: Queueing Network Models of Computer System Performance," ACM Computing Surveys, Vol. 10, #3, September 1978, pp 219-224.

38. Hamming, R.W. "Numerical Methods for Scientists and Engineers," 2nd ed. New York: Mc Graw-Hill, 1973.

39. Heidelberger, P. and Trivedi, K.S. "Analytic Queueing Models for Parallel Processing with Asynchronous Tasks," IEEE Transactions on Computers, Vol. c-31, No. 11, November 1982.

40. Heidelberger, P. and Trivedi, K.S. "Analytic Queueing Models for Programs with Internal Concurrency," IEEE Transactions on Computers, Vol. c-31, No. 11, January 1983.

41. Herzog, U., Woo, L.S., and Chandy, K.M., "Solution of Queueing Problems by a Recursive Technique," IBM J. of Research and Development 19, 3 (May 1975) pp. 295-300.

42. Jackson, J.R. "Jobshop Like Queueing Systems," Management Science 10(1963), pp 131-142.

43. Kleinrock, L. "Queueing Systems I," John Wiley, New York, 1975

44. Kleinrock, L. "Queueing Systems II," John Wiley, New York, 1976

45. Kobayashi, H., Modeling and Analysis: An Introduction to System Performance Evaluation Methodology, Addison-Wesley, Reading, MA (1978).

46. Lam,C.Y., "Data Storage Hierarchy Systems for Database Computers," Tech Rep #4, August 1979, MIT Sloan School.

47. Lam, C.Y. and Madnick, S.E., "Properties of Storage Hierarchy Systems with Multiple Page Sizes and Redundant Data," ACM Transactions on Database Systems, Vol. 4, No. 3, September 1979, pp 345-367.

48. Lam, S.S. and Shanker, A.U, " Response Time Distribution for a Multi-class Queue with Feedback," ACM 1980 0-89791-019-2, pp. 225-243.

49. Lavenberg, S.S. "Computer Performance Modeling Handbook," Academic Press, 1983

50. Lavenberg, S.S. and Slutz, D.R., "Introduction to Regenerative Simulation," IBM J. of Research and Development 19, (September 1975) pp. 458-463.

51. Lavenberg, S.S. and Sauer, C.H., "Sequential Stopping Rules for the Regenerative Method of Simulation," IBM J. of Research and Development 21, (November 1977) pp. 545-558.

52 Lazowska, E.D., "The Benchmarking, Tunign and Analytic Modelling of VAX/VMS," Dept. of Computer Science, U. of Washington, Seattle, Tech. Rep. 79-04-01, April, 1979.

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

54. Lucas, H.C., Jr., "Performance Evaluation and Monitoring," Computing Surveys, vol. 3, No. 3, September, 1971.

55. Madnick, S.E., "Storage Hierarchy Systems," Report No. TR-105, Project MAC, MIT, Cambridge, MA, 1973.

56. Madnick, S.E., "Trends in Computers and Computing: The Information Utility," Science, Vol. 185, March 1977, pp 1191-1199.

57. Madnick, S.E., "The INFOPLEX Database Computer, Concepts and Directions," Proc. IEEE Comp. Con., February 1979, pp 168-176.

58. Madnick, S.E., Research meeting with the author.

59. Maekawa, M and Boyd, D.L. "Two Models of Task Overlap with Jobs of Multiprocessing Multiprogramming Systems," Proc. 1976 Int. Conf. on Parallel Processing, Detroit, August 1976, pp 83-91.

60. Marie, R.A., "An Approximate Analytical Method for General Queueing Networks," IEEE Transactions on Software Engineering SE-5, 5 (September 1979).

61. Muntz, R.R. "Poisson Departure Processes and Queueing Networks," IBM Res. Rep. RC-4145, 1972.

62. Peterson, M. and Bulgren, W. "Studies in Markov Models of Computer Systems," Proc. 1975 ACM Annual Conf., Minneapolis, Minn., pp 102-107.

63. Price, T.G. "Models of Multiprogrammed Computer Systems with I/O buffering," Proc. 4th Texas Conf. Comput. Syst., Austin, 1975.

64. Reiser, M. and Kobayashi, H. "Queueing Networks with Multiple Closed Chains: Theory and Computational Algorithms," IBM J. Res. Develop., May 1975, pp 283-294.

65. Reiser, M., "Numerical Methods in Separable Queueing Networks," IBM Research Report RC-5842, Yorktown Heights, NY (February 1976).

66. Reiser, M. and Chandy, K.M. "The Impact of Distributions and Disciplines on Multiple Processor Systems," CACM, Vol.22, pp 25-34, 1979.

67. Reiser, M. and Lavenberg, S.S., "Mean Value Analysis of Closed Multichain Queueing Networks," IBM Research Report RC-7023, Yorktown heights, NY (March 1978). JACM 27, 2 (April 1980) pp. 313-322.

68. Reiser, M. and Sauer, C.H., "Queueing Network Models: Methods of Solution and their Program Implementation," in K.M. Chandy and R.T. Yeh, editors, Current Trends in Programming Methodology, Vol. III: Software Modeling and Its Impact on Performance. Prentice-Hall (1978) pp. 115-167.

69. Reiser, M., "Mean Value Analysis and Convolution Method for Queue-Dependent Servers in Closed Queueing Networks," to appear as an IBM Research Report (Zurich).

70. Reiser, M., "Numerical Methods in Separable Queueing Networks," IBM Research Report RC-5842, Yorktown Heights, NY (February 1976).

71. Sauer, C.H. and Chandy, K.M. "Computer Systems Performance Modeling," Printice-Hall, Englewood Cliffs, New Jersey, 1981.

72. Sauer, C.H. and Chandy, K.M., "Approximate Analysis of Central Server Models", IBM J. of Research and Development 19, 3 (May 1975) pp. 301-313.

73. Sauer, C.H. and Chandy, K.M., "Approximate Solutions of Queueing Models", IEEE Computer, April 1980, pp.25-32.

74. Sauer, C.H., WOO, L.S. ans Chang, W., "Hybrid Analysis/Simulation: Distributed Networks," RC-6341, IBM Research, Yorktown Heights, NY (June 1976).

75. Sauer, C.H., "Confidence Intervals for Queueing Simulations of Computer Systems," RC-6669, IBM Research, Yorktown Heights, NY (July 1977).

76. Sauer, C.H. and MacNair, E.A., "Computer/Communication System Modeling with Extended Queueing Networks," RC-6654, IBM Research, Yorktown Heights, NY (July 1977).

77. Sauer, C.H. and MacNair, E.A., "Queueing Network Software for Systems Modeling," RC-7143, IBM Research, Yorktown, NY (May 1978). Software Practice and Experience 9, 5 (May 1979).

78. Sauer, C.H., MacNair, E.A. and Salza, S., "A Language for Extended Queueing Networks," IBM Resesarch Report RC-7996, December 1979. IBM J. of Research and Development 24, 6 (November 1980).

79. Sauer, C.H., MacNair, E.A., and Kurose, J.F. "The Research Queueing Package Version 2: Introduction and Examples," RA 138, 4/12/82 IBM Thomas J. Watson Research Center, Yorktown Heights, New york 10598.

80. Sevcik, K.C., "Priority Scheduling Disciplines in Queueing Network Models of Computer Systems," 1977 IFIP Congress Proceedings pp. 565-570.

81. Sevcik, A., Levy, S.K., Tripathi, S.K. and Zahorjan, J.L., "Improved Approximations of Aggregated Queueing Network Subsystems," Computer Performance, K.M. Chandy and M. Reiser (Eds.), Elsevier North-Holland, Inc., New York, 1977, pp.1-22.

82. Sevcik, K.C. and Klawe, M.M., "Operational Analysis Versus Stochastic Modelling of Computer Systems," Proc. Computer Science and Statistics: 12th Annual Symposium on the Interface, University of Waterloo, May 1979.

83. Sevcik, K.C. and Mitrani, "The Distribution of Queueing Network States At Input and Output Instants,"

84. Sherman, S.W., Baskett, F. and Browne, J.C., "Trace Driven Modeling and Analysis of CPU Scheduling in a Multiprogramming System," CACM 15, (1972) pp. 1063-1069.

85. Shum, A. and Buzen, J.P., "The EPF Technique: A Method for Obtaining Approximate Solutions to Closed Queueing Networks with General Service Times," Measuring Modeling and Evaluating computer Systems, Beilner, H. and Gelenbe, E. (Eds.) North-Holland, Amsterdam (1977) pp. 201-220.

86. Suri, R. "Robustness of Queueing Network Formulas," JACM, Vol. 30, No.3, July 1983, pp.564-594.

87. Towsley, D., Chandy, K.M., and Browne, J.C. "Models for Parallel Processing within Programs: Applications to CPU:I/O and I/O:I/O Overlap," CACM, Vol.21, pp 821-831, 1978.

88. Towsley, D.F., "Local Balance Models of Computer Systems," Ph.D. Thesis, Univ. of Texas at Austin (Dec. 1975).

89. Towsley, D.F., "Queueing Network Models with State-Dependent Routing," JACM 27, 2 (April 1980) pp. 323-337.

90. Trivedi, K.S., "Analytic Modeling of Computer Systems," IEEE Computer, 1978, pp.38-52.

91. Trivedi, K.S. and Sigmon, T.M., "Optimal Design of Linear Storage Hierarchies," JACM, Vol.28, No. 2, April 1981, pp. 270-288.

92. Vantiborgh, H.T., "Near-Complete Decomposability of Queueing Networks with Clusters of Strongly Interacting Servers," ACM 1980, pp.81-92.

93. Wang, Y.R. and Madnick, S.E. "Performance Evaluation of the INFOPLEX Database Computer," Tech. Rep. #13, MIT Sloan School, April 1981.

94. Wang, Y.R. and Madnick, S.E. "Performance Evaluation of Distributed Systems with Unbalanced Flows," Tech. Rep. #14, MIT Sloan School, April 1983.

95. Wong, J.W. "Queueing Network Modeling of Computer Communication Networks," Computing Surveys, Vol. 10, No. 3, September 1978.

Appendix I:

Listing of Simulation program of Iterative Algorithms

This simulation program simulates closed networks with different populations and different workloads for the main chain and the UAP chain. The simulated network parameters are fed into the BI, BI/A, and BBS algorithms to test the algorithms' validity and efficiency. The program was written in BASICA on the IBM PC under DOS2.0.

```
100 DIM
       A1$(5)
110 DIM
       G(30),
       VSU(30),
       VSM(30),
       INFLATED.VSM(30)
120 DIM
       CASE.TABLE(400),
       ITERATION.TABLE(400,2)
130 LPRINT
       "========================= START SIMULATION .... ==================
          ==========="
140 LPRINT " "
150 INPUT "NUMBER OF EXPERIMENTS TO SIMULATE?",NUMBER.OF.EXPERIMENTS
160 LPRINT
       "FIVE ROUNDS OF SIMULATIONS, THE NUMBER OF EXPERIMENTS PER ROUND IS "
          ;NUMBER.OF.EXPERIMENTS
170 INPUT "RANDOM NUMBER SEED?",RANDOM.NUMBER.SEED
180 LPRINT "RONDOM NUMBER SEED IS ";RANDOM.NUMBER.SEED
190 LPRINT " "
200 RANDOMIZE(RANDOM.NUMBER.SEED)
210 CASE.TYPE = 0
220 RELATIVE.ERROR = .001
230 FOR ROUND = 1 TO 5 :

       REM 5 INDEPENDENT SIMULATIONS TO RUN

240    EXPERIMENT.NUMBER = 1
250    WHILE EXPERIMENT.NUMBER<=NUMBER.OF.EXPERIMENTS
260       PRINT " "
270       PRINT
             "========================== START EXPERIMENT ";
                EXPERIMENT.NUMBER;" =============================="
280       PRINT " "
290       MAX.VSU=0:
          NUMBER.OF.ITERATIONS=0:
          LOWER.BOUND=0:
          UPPER.BOUND=0
300       NUMBER.OF.FACILITIES = INT(RND*19) + 2
310       NUMBER.OF.CUSTOMERS = INT(RND*20) + 1
320       PRINT
             "NUMBER.OF.CUSTOMERS= ";NUMBER.OF.CUSTOMERS;TAB(40);
                "NUMBER.OF.FACILITIES= ";NUMBER.OF.FACILITIES
330       VSM.INDEX = 0
340       FOR M = 1 TO NUMBER.OF.FACILITIES
350          VSM(M) = INT(RND*6) * RND
360          VSU(M) = INT(RND*4) * RND
370          PRINT "VSM(";M;")= ";VSM(M);TAB(40);"VSU(";M;")= ";VSU(M)
380          IF
                VSM(M)>0
                   THEN
                      VSM.INDEX = 1
390          IF
```

```
              VSU(M)> MAX.VSU
                 THEN
                    MAX.VSU = VSU(M):
                    MAX.VSU.INDEX = M
400     NEXT M
410     PRINT "MAX.VSU= ";MAX.VSU;TAB(40);"MAX.VSU.INDEX= ";MAX.VSU.INDEX
420     PRINT " "
1000    PRINT
           "============ START STABILITY CONDITION TEST TO IDENTIFY CASE.T
              YPE ============"
1010    PRINT " "
1020    IF
           MAX.VSU = 0 OR VSM.INDEX = 0
              THEN
                 EXPERIMENT.NUMBER = EXPERIMENT.NUMBER - 1:
                 GOTO 3200
1030    MAX.XM =1/MAX.VSU
1040    PRINT "MAX.XM= ";MAX.XM
1050    X.EST =0
1060    GOSUB 4000
1065    MVI=MAX.VSU.INDEX
1070    IF
           XM<MAX.XM
              THEN
                 C=1:
                 U=XM:
                 L=0:
                 FL=XM:
                 X.EST=XM:
                 GOSUB 4000:
                 FU=XM
              ELSE
                 IF
                    VSM(MVI)>0
                       THEN
                          C=2:
                          U=MAX.XM:
                          L=0:
                          FL=XM:
                          FU=0
                       ELSE
                          X.EST=MAX.XM:
                          GOSUB 4000:
                          IF
                             XM<=MAX.XM
                                THEN
                                   C=3:
                                   U=MAX.XM:
                                   L=XM:
                                   FU=0:
                                   X.EST=XM:
                                   GOSUB 4000:
                                   FL=XM
                                ELSE
```

```
                                    CASE.TYPE=4:
                                    GOTO 3120
1080  CASE.TYPE=C:
      INITIAL.UPPER.BOUND=U:
      INITIAL.LOWER.BOUND=L:
      F.LOWER.BOUND=FL:
      F.UPPER.BOUND=FU
-1090 PRINT
         "THE CASE.TYPE OF EXPERIMENT ";EXPERIMENT.NUMBER;" IS ";
            CASE.TYPE
1100  PRINT " "
2000  PRINT
         "============= START <BOUNDED INTERPOLATION> ALGORITHM WITH TRA
            CE ============="
2010  PRINT " "
2020  UPPER.BOUND = INITIAL.UPPER.BOUND:
      LOWER.BOUND = INITIAL.LOWER.BOUND
2030  SLOPE = (F.LOWER.BOUND - F.UPPER.BOUND)/(UPPER.BOUND -
         LOWER.BOUND)
            :
         DELTA=(UPPER.BOUND-F.UPPER.BOUND)/(1+SLOPE):
         X.EST = UPPER.BOUND - DELTA
2040  GOSUB 4000
2050  NUMBER.OF.ITERATIONS = 1
2060  WHILE ( ABS(XM - X.EST) / X.EST ) > RELATIVE.ERROR
2070     IF
            XM<LOWER.BOUND
               THEN
                  LAST.X.EST=LOWER.BOUND:
                  LAST.XM=F.LOWER.BOUND:
                  UPPER.BOUND=X.EST:
                  F.UPPER.BOUND=XM
               ELSE
                  IF
                     UPPER.BOUND<XM
                        THEN
                           LAST.X.EST=UPPER.BOUND:
                           LAST.XM=F.UPPER.BOUND:
                           LOWER.BOUND=X.EST:
                           F.LOWER.BOUND=XM
2080     IF
            LOWER.BOUND<=XM AND XM <= UPPER.BOUND
               THEN
                  IF
                     XM<=X.EST
                        THEN
                           LAST.X.EST=LOWER.BOUND:
                           LAST.XM=F.LOWER.BOUND:
                           UPPER.BOUND=X.EST:
                           F.UPPER.BOUND=XM
                        ELSE
                           LAST.X.EST=UPPER.BOUND:
                           LAST.XM=F.UPPER.BOUND:
                           LOWER.BOUND=X.EST:
```

```
                         F.LOWER.BOUND=XM
2090      SLOPE=(LAST.XM-XM)/(X.EST-LAST.X.EST):
          DELTA=(X.EST-XM)/(SLOPE+1)
2100      PRINT
             "DELTA";TAB(15);"F.LOWER.BOUND";TAB(30);"F.UPPER.BOUND";TAB(
                46);"LAST.X.EST";TAB(61);"LAST.XM"
2110      PRINT
             DELTA;TAB(15);F.LOWER.BOUND;TAB(30);F.UPPER.BOUND;TAB(45);
                LAST.X.EST;TAB(60);LAST.XM
2120      PRINT " "
2130      X.EST = X.EST-DELTA
2140      GOSUB 4000
2150      NUMBER.OF.ITERATIONS = NUMBER.OF.ITERATIONS + 1
2160   WEND
2170   ITERATION.TABLE(EXPERIMENT.NUMBER,1)= NUMBER.OF.ITERATIONS
2180   PRINT
          "CASE.TYPE:";CASE.TYPE;"; NUMBER.OF.ITERATIONS:";
             NUMBER.OF.ITERATIONS;"; FINAL X.ESTIMATE:";X.EST
2190   PRINT " "
3000   PRINT
          "============= START [BOUNDED BINARY SEARCH] ALGORITHM WITH TRA
             CE ============="
3010   PRINT " "
3020   UPPER.BOUND = INITIAL.UPPER.BOUND:
       LOWER.BOUND = INITIAL.LOWER.BOUND
3030   X.EST = (UPPER.BOUND + LOWER.BOUND)/2
3040   GOSUB 4000
3050   NUMBER.OF.ITERATIONS = 1
3060   WHILE ( ABS(XM - X.EST) / X.EST ) > RELATIVE.ERROR
3070      IF
             XM<LOWER.BOUND
                THEN
                   UPPER.BOUND=X.EST
                ELSE
                   IF
                      LOWER.BOUND<=XM AND XM <= UPPER.BOUND
                         THEN
                            IF
                               XM<=X.EST
                                  THEN
                                     LOWER.BOUND=XM:
                                     UPPER.BOUND=X.EST
                                  ELSE
                                     LOWER.BOUND=X.EST:
                                     UPPER.BOUND=XM
                         ELSE
                            LOWER.BOUND=X.EST
3080      X.EST =(LOWER.BOUND+UPPER.BOUND)/2
3090      GOSUB 4000
3100      NUMBER.OF.ITERATIONS = NUMBER.OF.ITERATIONS + 1
3110   WEND
3120   ITERATION.TABLE(EXPERIMENT.NUMBER,2)= NUMBER.OF.ITERATIONS
3130   CASE.TABLE(EXPERIMENT.NUMBER) = CASE.TYPE
3140   PRINT " "
```

```
3150  |   |   PRINT
      |   |      "CASE.TYPE:";CASE.TYPE;"; NUMBER.OF.ITERATIONS:";
      |   |         NUMBER.OF.ITERATIONS;"; FINAL X.ESTIMATE:";X.EST
3160  |   |   PRINT
      |   |      "=================== END OF EXPERIMENT ";EXPERIMENT.NUMBER;
      |   |         " ========================="
3170  |   |   PRINT " "
3180  |   |   PRINT " "
3190  |   |   EXPERIMENT.NUMBER = EXPERIMENT.NUMBER + 1
3200  |   WEND
3210  |   GOSUB 4170
3220  |   PRINT " "
3230  |   PRINT
      |      "============================= END OF SIMULATION ================
      |         ==============="
3240  |   LPRINT " "
3250  |   LPRINT
      |      "================================================================
      |         ==============="
3260  |   LPRINT " "
3270  |   LPRINT "RANDOM.NUMBER.SEED FOR ROUND ";ROUND+1;" IS ";RND
3280  |   LPRINT " "
3290  NEXT ROUND
3300  STOP
4000  FOR M=1 TO NUMBER.OF.FACILITIES
4010  |   IF
      |      VSM(M)>0
      |         THEN
      |            INFLATED.VSM(M) = VSM(M)/(1-VSU(M)*X.EST)
      |         ELSE
      |            INFLATED.VSM(M) = 0
4020  NEXT M
4030  FOR N = 1 TO NUMBER.OF.CUSTOMERS
4040  |   G(N)=0
4050  NEXT N
4060  G(0) = 1
4070  FOR M = 1 TO NUMBER.OF.FACILITIES
4080  |   FOR N=1 TO NUMBER.OF.CUSTOMERS
4090  |   |   G(N)=G(N)+INFLATED.VSM(M)*G(N-1)
4100  |   NEXT N
4110  NEXT M
4120  XM =G(NUMBER.OF.CUSTOMERS-1)/G(NUMBER.OF.CUSTOMERS)
4130  PRINT "LOWER.BOUND";TAB(17);"UPPER.BOUND";TAB(31);"X.EST";TAB(46);"XM"
4140  PRINT LOWER.BOUND;TAB(15);UPPER.BOUND;TAB(30);X.EST;TAB(45);XM
4150  PRINT " "
4160  RETURN
4170  PRINT " "
5000  LPRINT
         "========================= STATISTICAL ANALYSIS  ==================
            ==========="
5010  LPRINT " "
5020  LPRINT "EXPERIMENT","CASE.TYPE","INTERPOLATE","BOUNDED.BINARY"
5030  INTERP.SUM = 0
5040  BINARY.SUM = 0
```

```
5050 FOR I = 1 TO NUMBER.OF.EXPERIMENTS
5060     LPRINT I,CASE.TABLE(I),ITERATION.TABLE(I,1),ITERATION.TABLE(I,2)
5070     INTERP.SUM = INTERP.SUM + ITERATION.TABLE(I,1)
5080     BINARY.SUM = BINARY.SUM + ITERATION.TABLE(I,2)
5090 NEXT I
5100 LPRINT "----------","---------","-----------","-------------"
5110 LPRINT "     ","TOTAL",INTERP.SUM,BINARY.SUM
5120 INTERP.MEAN=INTERP.SUM/NUMBER.OF.EXPERIMENTS:
     BINARY.MEAN=BINARY.SUM/NUMBER.OF.EXPERIMENTS
5130 LPRINT "     ","MEAN   ",INTERP.MEAN,BINARY.MEAN
5140 INTERP.SD=0:
     BINARY.SD=0
5150 FOR J=1 TO NUMBER.OF.EXPERIMENTS
5160     D=ITERATION.TABLE(J,1)-INTERP.MEAN:
         INTERP.SD=INTERP.SD+D*D:
         D=ITERATION.TABLE(J,2)-BINARY.MEAN:
         BINARY.SD=BINARY.SD+D*D
5170 NEXT J
5180 LPRINT
         "   ","S.D.   ",SQR(INTERP.SD/NUMBER.OF.EXPERIMENTS),SQR(BINARY.SD/
             NUMBER.OF.EXPERIMENTS)
5190 A1$(1)= "XM(XM=0)<=MAX.XM"
5200 A1$(2)= "XM(XM=0)>MAX.XM, AND VSM(MAXVSU)>0"
5210 A1$(3)="XM(XM=0)>MAX.XM, VSM(MAX.VSU)=0, AND XM(MAX.XM)<MAX.XM"
5220 A1$(4)="XM(XM=0)>MAX.XM, VSM(MAX.VSU)=0, AND XM(MAX.XM)>=MAX.XM"
5230 FOR CASE.TYPE = 1 TO 4
5240     LPRINT " "
5250     LPRINT
             "=============================================================
                 =============="
5260     LPRINT " "
5270     LPRINT "==== CASE.TYPE ";CASE.TYPE;":";A1$(CASE.TYPE);" ===="
5280     LPRINT " "
5290     INTERP.TYPE.ITERATIONS=0
5300     BINARY.TYPE.ITERATIONS=0
5310     NUMBER.OF.TYPE.EXPERIMENTS=0
5320     LPRINT "ITERATIONS","INTERPOLATE","BOUNDED.BINARY"
5330     FOR NUMBER.OF.ITERATIONS = 1 TO 25
5340         INTERP.TYPE.EXPERIMETNS=0
5350         BINARY.TYPE.EXPERIMENTS=0
5360         FOR J=1 TO NUMBER.OF.EXPERIMENTS
5370             IF
                     CASE.TABLE(J)=CASE.TYPE
                         THEN
                             IF
                                 ITERATION.TABLE(J,1)=NUMBER.OF.ITERATIONS
                                     THEN
                                         INTERP.TYPE.EXPERIMETNS=
                                             INTERP.TYPE.EXPERIMETNS+1
                                     ELSE
                                         IF
                                             ITERATION.TABLE(J,2)=NUMBER.OF.ITERATIONS
                                                 THEN
                                                     BINARY.TYPE.EXPERIMENTS=
```

```
                                           BINARY.TYPE.EXPERIMENTS+1
5380      NEXT J
5390      IF
             INTERP.TYPE.EXPERIMETNS=0 AND BINARY.TYPE.EXPERIMENTS=0
                THEN
                   GOTO 5440
5400      LPRINT
             NUMBER.OF.ITERATIONS,INTERP.TYPE.EXPERIMETNS,
                BINARY.TYPE.EXPERIMENTS
5410      INTERP.TYPE.ITERATIONS=INTERP.TYPE.ITERATIONS+
             INTERP.TYPE.EXPERIMETNS*NUMBER.OF.ITERATIONS
5420      BINARY.TYPE.ITERATIONS=BINARY.TYPE.ITERATIONS+
             BINARY.TYPE.EXPERIMENTS*NUMBER.OF.ITERATIONS
5430      NUMBER.OF.TYPE.EXPERIMENTS=NUMBER.OF.TYPE.EXPERIMENTS+
             INTERP.TYPE.EXPERIMETNS
5440   NEXT NUMBER.OF.ITERATIONS
5450   LPRINT "----------","-----------","-------------"
5460   LPRINT "TOTAL",INTERP.TYPE.ITERATIONS,BINARY.TYPE.ITERATIONS
5470   LPRINT
          "REPLICATIONS",NUMBER.OF.TYPE.EXPERIMENTS,
             NUMBER.OF.TYPE.EXPERIMENTS
5480   IF
          NUMBER.OF.TYPE.EXPERIMENTS=0
             THEN
                GOTO 5560
5490   INTERP.TYPE.MEAN=INTERP.TYPE.ITERATIONS/NUMBER.OF.TYPE.EXPERIMENTS:
       BINARY.TYPE.MEAN=BINARY.TYPE.ITERATIONS/NUMBER.OF.TYPE.EXPERIMENTS
5500   LPRINT "MEAN",INTERP.TYPE.MEAN,BINARY.TYPE.MEAN
5510   INTERP.TYPE.SD=0:
       BINARY.TYPE.SD=0
5520   FOR J=1 TO NUMBER.OF.EXPERIMENTS
5530      IF
             CASE.TABLE(J)=CASE.TYPE
                THEN
                   D=ITERATION.TABLE(J,1)-INTERP.TYPE.MEAN:
                   INTERP.TYPE.SD=INTERP.TYPE.SD+D*D:
                   D=ITERATION.TABLE(J,2)-BINARY.TYPE.MEAN:
                   BINARY.TYPE.SD=BINARY.TYPE.SD+D*D
5540   NEXT J
5550   LPRINT
          "S.D.   ",SQR(INTERP.TYPE.SD/NUMBER.OF.TYPE.EXPERIMENTS),SQR(
             BINARY.TYPE.SD/NUMBER.OF.TYPE.EXPERIMENTS)
5560 NEXT CASE.TYPE
5570 RETURN
```

Appendix II:

Listing of Sample Audit Output

This sample audit output is generated by TAD for the P1L3 model documented in Chapter VI.3.2. It enables designers to study the behavior of the distributed control algorithms.

ENTER A LOCALITY(ASSUME THE SAME ACROSS LEVELS)!
.7

ENTER READ%!
.7

CHECK IN DSH LEVEL ONE PE.

NUMBER OF FACILITIES		LEVEL	VISIT-RATIO	SERVICE-TIME	VS-PRODUCT	CHAIN-TYPE
1	PE	1	.70000	200.000	140.0	1

READ-THROUGH-MESSAGE STOPS WHEN DATA IS FOUND;
IT'S FOLLOWED BY READ-THROUGH-RESULT-FOUND TRANSACTION.

READ-THROUGH-MSG.

NUMBER OF FACILITIES		LEVEL	VISIT-RATIO	SERVICE-TIME	VS-PRODUCT	CHAIN-TYPE
1	LBUS	1	.21000	100.000	21.0	1
1	GC	1	.21000	100.000	21.0	1
1	GBUS	1	.21000	100.000	21.0	1
1	GC	2	.21000	100.000	21.0	1
1	LBUS	2	.21000	100.000	21.0	1
1	PE	2	.21000	200.000	42.0	1
1	LBUS	2	.06300	100.000	6.3	1
1	GC	2	.06300	100.000	6.3	1
1	GBUS	2	.06300	100.000	6.3	1
1	GC	3	.06300	100.000	6.3	1
1	LBUS	3	.06300	100.000	6.3	1
1	PE	3	.06300	200.000	12.6	1

READ-THROUGH-RESULTS FOUND AT LEVEL 1

NUMBER OF FACILITIES		LEVEL	VISIT-RATIO	SERVICE-TIME	VS-PRODUCT	CHAIN-TYPE
1	PE	1	.49000	100.000	49.0	1

READ-THROUGH-RESULTS FOUND AT LEVEL 2

NUMBER OF FACILITIES		LEVEL	VISIT-RATIO	SERVICE-TIME	VS-PRODUCT	CHAIN-TYPE
1	LBUS	2	.14700	100.000	14.7	1
2	LSS	2	.14700	1000.000	73.5	1
1	LBUS	2	.14700	100.000	14.7	1
1	GC	2	.14700	100.000	14.7	1
1	GBUS	1	.14700	100.000	14.7	1

TAKE CARE OF LEVEL 1 UP TO LEVEL 1 BROADCAST.

NUMBER OF FACILITIES		LEVEL	VISIT-RATIO	SERVICE-TIME	VS-PRODUCT	CHAIN-TYPE
1	GC	1	.14700	100.000	14.7	1
1	LBUS	1	.14700	100.000	14.7	1
1	PE	1	.14700	100.000	14.7	1

OVERFLOW FROM LEVEL 2 BROADCAST.

NUMBER OF FACILITIES		LEVEL	VISIT-RATIO	SERVICE-TIME	VS-PRODUCT	CHAIN-TYPE
1	LBUS	1	.07350	100.000	7.4	2
1	GC	1	.07350	100.000	7.4	2
1	GBUS	1	.07350	100.000	7.4	2
1	GC	2	.07350	100.000	7.4	2
1	LBUS	2	.07350	100.000	7.4	2
1	PE	2	.07350	200.000	14.7	2

READ-THROUGH-RESULTS FOUND AT LEVEL 3

NUMBER OF FACILITIES		LEVEL	VISIT-RATIO	SERVICE-TIME	VS-PRODUCT	CHAIN-TYPE
1	LBUS	3	.06300	100.000	6.3	1
2	LSS	3	.06300	2000.000	63.0	1

1	LBUS	3	.06300	800.000	50.4	1
1	GC	3	.06300	100.000	6.3	1
1	GBUS	2	.06300	800.000	50.4	1

TAKE CARE OF LEVEL 1 UP TO LEVEL 2 BROADCAST.

NUMBER OF FACILITIES		LEVEL	VISIT-RATIO	SERVICE-TIME	VS-PRODUCT	CHAIN-TYPE
1	GC	1	.06300	100.000	6.3	1
1	LBUS	1	.06300	100.000	6.3	1
1	PE	1	.06300	100.000	6.3	1
1	GC	2	.06300	100.000	6.3	2
1	LBUS	2	.06300	800.000	50.4	2
1	PE	2	.06300	200.000	12.6	2
1	LBUS	2	.06300	800.000	50.4	2
2	LSS	2	.06300	1000.000	31.5	2

OVERFLOW FROM LEVEL 3 BROADCAST.

NUMBER OF FACILITIES		LEVEL	VISIT-RATIO	SERVICE-TIME	VS-PRODUCT	CHAIN-TYPE
1	LBUS	1	.03150	100.000	3.2	2
1	GC	1	.03150	100.000	3.2	2
1	GBUS	1	.03150	100.000	3.2	2
1	GC	2	.03150	100.000	3.2	2
1	LBUS	2	.03150	100.000	3.2	2
1	PE	2	.03150	200.000	6.3	2
1	LBUS	2	.03150	100.000	3.2	2
1	GC	2	.03150	100.000	3.2	2
1	GBUS	2	.03150	100.000	3.2	2
1	GC	3	.03150	100.000	3.2	2
1	LBUS	3	.03150	100.000	3.2	2
1	PE	3	.03150	200.000	6.3	2

STB TRANSACTION.

NUMBER OF FACILITIES		LEVEL	VISIT-RATIO	SERVICE-TIME	VS-PRODUCT	CHAIN-TYPE

1	PE	1	.30000	100.000	30.0	1
1	LBUS	1	.30000	100.000	30.0	2
1	GC	1	.30000	100.000	30.0	2
1	GBUS	1	.30000	100.000	30.0	2
1	GC	2	.30000	100.000	30.0	2
1	LBUS	2	.30000	100.000	30.0	2
1	PE	2	.30000	200.000	60.0	2
1	LBUS	2	.30000	100.000	30.0	2
2	LSS	2	.30000	1000.000	150.0	2
1	LBUS	2	.30000	800.000	240.0	2
1	GC	2	.30000	100.000	30.0	2
1	GBUS	2	.30000	800.000	240.0	2
1	GC	3	.30000	100.000	30.0	2
1	LBUS	3	.30000	800.000	240.0	2
1	PE	3	.30000	200.000	60.0	2
1	LBUS	3	.30000	800.000	240.0	2
2	LSS	3	.30000	2000.000	300.0	2

ACK TRANSACTION.

NUMBER OF FACILITIES		LEVEL	VISIT-RATIO	SERVICE-TIME	VS-PRODUCT	CHAIN-TYPE
1	LBUS	2	.30000	100.000	30.0	2
1	GC	2	.30000	100.000	30.0	2
1	GBUS	2	.30000	100.000	30.0	2
1	GC	1	.30000	100.000	30.0	2
1	LBUS	1	.30000	100.000	30.0	2
1	PE	1	.30000	200.000	60.0	2
1	LBUS	2	.30000	100.000	30.0	2
1	GC	2	.30000	100.000	30.0	2
1	GBUS	2	.30000	100.000	30.0	2
1	GC	1	.30000	100.000	30.0	2
1	LBUS	1	.30000	100.000	30.0	2
1	PE	1	.30000	200.000	60.0	2
1	LBUS	3	.30000	100.000	30.0	2
1	GC	3	.30000	100.000	30.0	2
1	GBUS	3	.30000	100.000	30.0	2
1	GC	2	.30000	100.000	30.0	2
1	LBUS	2	.30000	100.000	30.0	2
1	PE	2	.30000	200.000	60.0	2

Appendix III: Listing of TAD

TAD (Technique for Architectural Design) is an analytic software tool designed to evaluate the performance of the INFOPLEX Data Storage Hierarchy. It is implemented in BASICV on the PRIME 850 at the Sloan School of Management, Massachusetts Institute of Technology. A sample session of TAD is available in Appendix VI.

```
100  REM SYSTEM MAP:
110  REM
120  REM **********************************
130  REM *                                *
140  REM *   MAIN PROGRAM:  1210-1920     *
150  REM *   VISIT RATIO:   1930-3140     *
160  REM *   PERFORMANCE:   3150-3530     *
170  REM *   ERROR HANDLER: 3540-4200     *
180  REM *   PRIMITIVES:    4210-11460    *
190  REM *                                *
200  REM **********************************

210  REM
220  REM CONFIGURATION PARAMETERS:
230  REM
240  REM ******************************************
250  REM *                                        *
260  REM *   C9(0,0): # OF LEVELS IN THE MODEL.   *
270  REM *   C9(1,0): # OF GBUS'S.                *
280  REM *   C9(2,0): READ%.                      *
290  REM *   C9(3.0): # OF SERVICE FACILITIES.    *
300  REM *   C9(4,0):                             *
310  REM *   C9(5,0): # OF LOCALITIES TO COMPUTE. *
320  REM *   C9(0,L): # OF LBUS'S AT LEVEL L.     *
330  REM *   C9(1,L): # OF PE'S AT LEVEL L.       *
340  REM *   C9(2,L): # OF LSS'S AT LEVEL L.      *
350  REM *   C9(3,L): # OF GC'S AT LEVEL L.       *
360  REM *   C9(4,L): PROB. OF OVERFLOW LEVEL L.  *
370  REM *   C9(5,L): LOCALITY AT LEVEL L.        *
380  REM *                                        *
390  REM ******************************************

400  REM

410  REM FACILITY INDICATORS:
420  REM
430  REM ******************************************************
440  REM *                                                    *
450  REM *   F9(0,0): STARTING INDEX FOR GBUS'S.              *
460  REM *   F9(0,L): STARTING INDEX FOR LBUS'S AT LEVEL L.   *
470  REM *   F9(1,L): STARTING INDEX FOR PE'S AT LEVEL L.     *
480  REM *   F9(2,L): STARTING INDEX FOR LSS AT LEVEL L.      *
490  REM *   F9(3,L): STARTING INDEX FOR GC AT LEVEL L.       *
500  REM *                                                    *
510  REM ******************************************************
520  REM
530  REM STRING AND NUMERIC VARIABLES:
540  REM
550  REM ******************************************************
560  REM *                                                    *
570  REM *   A,B,C,D:   ARGUMENTS FOR LOOP MACROS.            *
580  REM *                                                    *
590  REM *   A7$(6,2):  FIGURE TITLE TEXT.                    *
600  REM *   A8$(5,2):  POLICY ALTERNATIVES.                  *
```

```
 610  REM *   A9$(0):     I/O FILE NAME.                            *
 620  REM *   A9$(1-5):   GLOBAL TEMPORARY VARIABLES.               *
 630  REM *   A9$(6):     "INVALID INPUT, PLEASE REENTER!"          *
 640  REM *   A9$(7):     USING FORMAT FOR VISIT-RATIO REPORT.      *
 650  REM *   A9$(8-11): "LBUS,PE,LSS,GC"                           *
 660  REM *                                                         *
 670  REM ***********************************************************
 680  REM
 690  REM ***********************************************************
 700  REM *                                                         *
 710  REM *   K9( 0): THE MAIN CHAIN RESPONSE TIME.                 *
 720  REM *   K9( 1): THE MAIN CHAIN THRUPUT.                       *
 730  REM *   K9( 2): THE UNBALANCED CHAIN THROUGHPUT.              *
 740  REM *   K9( 3):           NOT USED.                           *
 750  REM *   K9( 4):           NOT USED.                           *
 760  REM *   K9( 5): CURRENT LEVEL TO COMPUTE S.F. INDECIES.       *
 770  REM *   K9( 6): VISIT-RATIO UP TO LAST LEVEL.                 *
 780  REM *   K9( 7): VISIT-RATIO AT THIS LEVEL.                    *
 790  REM *   K9( 8): VISIT-RATIO DUMP FLAG.                        *
 800  REM *   K9( 9): TYPE OF S.F.                                  *
 810  REM *   K9(10): # OF TIMES OF VISITS TO A TYPE OF S.F.        *
 820  REM *   K9(11):           NOT USED.                           *
 830  REM *   K9(12):           NOT USED.                           *
 840  REM *   K9(13): MAXIMUM UTILIZATION OF AN OPEN SYSTEM.        *
 850  REM *   K9(14): POPULATION OF THE CLOSED CHAIN.               *
 860  REM *   K9(15): MAXIMUM POPULATION OF THE CLOSED CHAIN.       *
 870  REM *   K9(16): MAXIMUM S.F.'S BY DIM.                        *
 880  REM *   K9(17): MAXIMUM # OF LEVELS BY DIM.                   *
 890  REM *   K9(18): TYPE OF DATA TO PRINT OUT.                    *
 900  REM *   K9(19): CURRENT COMBINATION OF POLICIES.              *
 910  REM *                                                         *
 920  REM ***********************************************************
 930  REM
 940  REM ***************************************************************
 950  REM *                                                             *
 960  REM *   S7$(1-9,7):    TEMPORARY LEVEL FRAMEWORK.                 *
 970  REM *   S7$(10-18,7):  PERMANENT LEVEL FRAMEWORK.                 *
 980  REM *   S7$(0,0):      RESERVED TEMPORARY VARIABLE.               *
 990  REM *   S8$(3,30):     TEXT FRAMEWORK FOR 6 TYPES PRINT OUT.      *
1000  REM *   S9(0,0):       BUS MSG SERVICE TIME.                      *
1010  REM *   S9(0,L):       G/LBUS DATA SERVICE TIME AT LEVEL L.       *
1020  REM *   S9(1,L):       PE SERVICE TIME AT LEVEL L.                *
1030  REM *   S9(2,1):       LM SERVICE TIME AT LEVEL 1.                *
1040  REM *   S9(2,L):       LSS SERVICE TIME AT LEVEL L.               *
1050  REM *   S9(3,L):       GC SERVICE TIME AT LEVEL L.                *
1060  REM *                                                             *
1070  REM ***************************************************************
1080  REM
1090  REM ***************************************************************
1100  REM *                                                             *
1110  REM *   V9(0,M):       HIGH PRIORITY MAIN PATH VS SUM.            *
1120  REM *   V9(1,M):       NORMAL MAIN PATH VS SUM.                   *
1130  REM *   V9(2,M):       NORMAL UNBALANCED PATH VS SUM.             *
1140  REM *   V9(3,M):       LOW PRIORITY UNBALANCED PATH VS SUM.       *
```

```
1150  REM *   V9(4,M):      S.F.'S VS SUM OR UTILIZATION.            *
1160  REM *   V9(5,M):      INFLATED CHAIN VS SUM.                   *
1170  REM *   V9(6,M):      NBAR OF THE INFLATED CHAIN.              *
1180  REM *                                                          *
1190  REM *********************************************************

1200  REM

1210  DIM
         A7$(6,2),
         A8$(5,2),
         A9$(11),
         C9(5,6),
         F8(3,5),
         F9(3,6),
         G(50)
1220  DIM
         K8(8,5),
         K9(19),
         S7$(18,7),
         S8$(3,30),
         S9(3,6),
         V8(5,2),
         V9(6,100)
1230  K9(15) = 50!MAXIMUM POPULATION SIZE
1240  K9(17) = 6!CURRENT MAX # OF LEVELS
1250  K9(16) = 100!CURRENT MAX FACILITY NUMBER
1260  DEF FNC1 (X) = INT(X/10000)
1270  DEF FNP1 (X) = INT((X MOD 10000)/1000)
1280  DEF FNR2 (X) = INT((X MOD 1000)/100)
1290  DEF FNC3 (X) = INT((X MOD 100)/10)
1300  DEF FNP4 (X) = X MOD 10
1310  PRINT LIN(2)
1320  PRINT "     ******************************************"
1330  PRINT "     ***                                    ***"
1340  PRINT "     ***      INFOPLEX TAD VERSION 1.0      ***"
1350  PRINT "     ***                                    ***"
1360  PRINT "     ***  A TOOL FOR ARCHITECTURAL DESIGN   ***"
1370  PRINT "     ***                                    ***"
1380  PRINT "     ***            NOVEMBER 1983           ***"
1390  PRINT "     ***                                    ***"
1400  PRINT "     ******************************************"
1410  PRINT LIN(1)
1420  GOSUB 7950 !INITIALIZATION.
1430  PRINT LIN(1)
1440  ON
         ERROR
            GOTO 3550
1450  INPUT "IS THIS A NEW MODEL? CONFIRM YES/NO: ";A9$(1)
1460  GOSUB 10180
      !CONVERT INPUT TO Y OR N OR NO-CHANGE.
1470  IF
         A9$(1)="Y"
            THEN
```

```
                GOSUB 6360
             ELSE
                IF
                   A9$(1)="N"
                      THEN
                         GOSUB 8960
                      ELSE
                         GOTO 1450
1480  GOSUB 9380
      !COMPUTE # OF SERVICE FACILITIES;
1490  GOSUB 1800
      !PRINT OUT MODEL PARAMETERS.
1500  ON
         ERROR
            GOTO 3580
1510  PRINT LIN(1)
1520  INPUT "DO YOU WANT TO SAVE THE MODEL? CONFIRM YES/NO: ";A9$(1)
1530  GOSUB 10180
      !CONVERT INPUT TO Y OR N OR NO-CHANGE.
1540  IF
         A9$(1)<>"Y" AND A9$(1)<>"N"
            THEN
               GOTO 1520
1550  IF
         A9$(1)="Y"
            THEN
               GOSUB 9180
               !SAVE THE MODEL
1560  ON
         ERROR
            GOTO 3610
1570  PRINT LIN(1)
1580  INPUT
         "DO YOU WANT TO AUDIT THE VISIT-RATIO REPORT? CONFIRM YES/NO: ";A9$(
            1)
1590  GOSUB 10180
      !CONVERT INPUT TO Y OR N OR NO-CHANGE.
1600  IF
         A9$(1)<>"Y" AND A9$(1)<>"N"
            THEN
               GOTO 1580
1610  IF
         A9$(1)="Y"
            THEN
               K9(8)=1
            ELSE
               K9(8)=0!VISIT RATIO REPORT FLAG ON
               IF
                  K9(8)=1!
1620  DEFINE FILE #2="TOUT1.0" !TAD OUTPUT
         FILE(COMBINATION,READ%,LOCALITY,RES.TIME,THRUPUT)
1630  ( THEN )
         GOSUB 4380
         !SELECT THE COMBINATION OF POLICIES
```

```
1640  GOSUB 10410
      !SET PARAMETERS FOR POINT/CURVE POLICIES, OPEN/CLOSED SYSTEMS;
1650  FOR A1 = 1 TO C9(5,0) !FOR NUMBER OF LOCALITIES TO COMPUTE MEASURES
1660  |  GOSUB 9580
      |  !SYSTEM RESET
1670  |  GOSUB 1940
      |  !COMPUTE SUMS OF (VISIT-RATIO)*(SERVICE TIME)
1680  |  GOSUB 3160
      |  !COMPUTE PERFORMANCE MEASURES
1690  |  GOSUB 10110
      |  !PRINT/FILE (COMBINATION,READ%,LOCALITY,RES.TIME,THRUPUT)
1700  NEXT A1
1710  PRINT LIN(1)
1720  PRINT"END OF SESSION!"
1730  ON
         ERROR
            GOTO 3610
1740  PRINT LIN(1)
1750  INPUT
         "DO YOU WANT TO CONTINUE ON OTHER COMBINATIONS OF POLICIES? CONFIRM Y
            ES/NO: ";A9$(1)
1760  GOSUB 10180
      !CONVERT INPUT TO Y OR N OR NO-CHANGE.
1770  IF
         A9$(1)<>"Y" AND A9$(1)<>"N"
            THEN
               GOTO 1750
1780  IF
         A9$(1)="Y"
            THEN
               GOTO 1630
            ELSE
               STOP

1790  REM PRINT THE MODEL PARAMETERS

1800  PRINT LIN(1) !AND INITIALIZE VISIT-RATIO/PERFORMANCE BUFFERS.
1810  PRINT "NUMBER OF SERVICE FACILITIES IS: ";C9(3,0)
1820  PRINT LIN(1)
1830  PRINT "LEVEL 1 LOCAL MEMORY SERVICE TIME IS: ";S9(2,1);" ns."
1840  PRINT LIN(1)
1850  PRINT "BUS MESSAGE SERVICE TIME IS: ";S9(0,0);" ns."
1860  K9(18) = 1
1870  GOSUB 4220
      !PRINT OUT THE MODEL WITH DATA
1880  PRINT LIN(1)
1890  FOR A1 = 1 TO C9(0,0)
1900  |  PRINT "THE PROBABILITY OF OVERFLOW LEVEL ";A1;" IS: ";C9(4,A1);"."
1910  NEXT A1
1920  RETURN

1930  REM SUMS OF (VISIT-RATIO)*(SERVICE-TIME) COMPUTATION ROUTINE
```

```
1940  K9(7) = C9(2,0) !READ %; THE INITIAL CURRENT LEVEL VISIT-RATIO.
1950  K9(5) = 1!CHECK LEVEL 1 PE TO SEE
      IF
         READ-DATA HIT.
1960  ( THEN )
         GOSUB 7470
         !COMPUTE FACILITY INDECIES FIRST.
1970  A9$(5)="CHECK IN DSH LEVEL ONE PE."
1980  IF
         K9(8)=1
            THEN
               GOSUB 10940
1990  A = 1!FACILITY TYPE IS PE.
2000  B = 1!IT IS LEVEL 1.
2010  C = 1! CHAIN TYPE IS THE MAIN CHAIN W/O PRIORITY.
2020  D = C9(2,0) !VISIT RATIO IS READ%0!
2030  GOSUB 7600
      !ADD THE SERVICE LOAD TO PE.
2040  IF
         K9(8)=0
            THEN
               GOTO 2140
2050  PRINT LIN(1)
2060  PRINT"READ-THROUGH-MESSAGE STOPS WHEN DATA IS FOUND;"
2070  PRINT"IT'S FOLLOWED BY READ-THROUGH-RESULT-FOUND TRANSACTION."
2080  PRINT LIN(1)
2090  A9$(5)="READ-THROUGH-MSG."
2100  GOSUB 10940

2110  REM READ-THROUGH-MSG: LBUS -> GC -> GBUS -> GC -> LBUS -> PE.


2120  REM READ-THROUGH-MSG TRANSACTION, STOPS WHEN FOUND(HIT).


2130  REM WHEN FOUND, STARTS READ-THROUGH-RESULTS-FOUND TRANSACTION.

2140  FOR B1 = 1 TO C9(0,0)-1
2150  |  K9(5) = B1
2160  |  GOSUB 7470
      |  !COMPUTE SERVICE FACILITY INDECIES.
2170  |  K9(7) = K9(7)*(1-C9(5,B1)) !MISSING CURRENT LEVEL.
2180  |  B = B1
2190  |  C = 1
2200  |  D = K9(7)
2210  |  A = 0
2220  |  GOSUB 7740
      |  !-> LBUS
2230  |  A = 3
2240  |  GOSUB 7600
      |  !-> GC
2250  |  GOSUB 7880
      |  !-> GBUS
2260  |  K9(5) = B1+1
```

```
2270 |  GOSUB 7470
2280 |  B = B1 + 1
2290 |  A = 3
2300 |  GOSUB 7600
     |  !-> GC
2310 |  A = 0
2320 |  GOSUB 7740
     |  !-> LBUS
2330 |  A = 1
2340 |  GOSUB 7600
     |  !-> PE
2350 NEXT B1

2360 REM READ-THROUGH-RESULTS-FOUND TRANSACTION.

2370 K9(7) = C9(2,0) !INITIAL CURRENT LEVEL VISIT-RATIO.
2380 FOR B1 = 1 TO C9(0,0) !READ-THROUGH-RESULTS-FOUND AT LEVEL B1.
2390 |  A9$(5)="READ-THROUGH-RESULTS FOUND AT LEVEL "+STR$(B1)
2400 |  IF
     |     K9(8)=1
     |        THEN
     |           GOSUB 10940
2410 |  K9(5) = B1 !CURRENT LEVEL
2420 |  GOSUB 7470
     |  !COMPUTE CURRENT LEVEL FACILITY INDECIES.
2430 |  K9(6) = K9(7) !CURRENT LEVEL VISIT-RATIO BECOMES LAST LEVEL.
2440 |  K9(7) = K9(6)*(1-C9(5,B1)) !MISS CURRENT LEVEL.
2450 |  B2 = K9(6)*C9(5,B1) !MISS UP TO LAST AND HIT CURRENT LEVEL.
2460 |  IF
     |     B1 > 1
     |        THEN
     |           GOTO 2600

2470 |  REM READ DATA FOUND AT LEVEL 1.

2480 |  A = 1!TYPE OF SERVICE FACILITY IS PE.
2490 |  B = 1!CURRENT LEVEL IS B1.
2500 |  C = 1! CHAIN TYPE IS MAIN CHAIN W/O PRIORITY.
2510 |  D = B2 !HIT THE FIRST LEVEL; VISIT-RATIO IS B2.
2520 |  B3 = S9(1,1) !SAVE PE1 SERVICE TIME.
2530 |  S9(1,1)=S9(2,1) !DATA SERVICE TIME INSTEAD OF DIRECTORY LOOK-UP
     |     TIME..
     |
2540 |  GOSUB 7600
     |  !LOOP MACRO FOR NON-GBUS SERVICE FACILITIES.
2550 |  S9(1,1) = B3 !RESTORE PE1 SERVICE TIME.
2560 |  GOTO 2840

2570 |  REM -> LBUS(MSG) -> LSS -> LBUS(DATA SIZE(B1-1)) -> GC -> GBUS --->
           BROADCAST.

2580 |  REM READ-THRU-RESULTS FOUND NOT AT LEVEL 1.
2590 |  REM TAKE CARE OF LEVEL B1.
```

```
2600    K9(5) = B1 !SET LEVEL.
2610    GOSUB 7470
        !COMPUTE FACILITY INDECIES FOR LEVEL B1.
2620    B = B1
2630    C = 1
2640    D = B2
2650    A = 0
2660    GOSUB 7740
        !LBUS MSG LOAD.
2670    A = 2!LSS
2680    GOSUB 7600
2690    A = 0
2700    GOSUB 7670
        !LBUS DATA(B1-1)
2710    A = 3
2720    GOSUB 7600 !GC

2730    REM TAKE CARE OF GBUS.

2740    B = B1-1!CURRENT LEVEL IS B1, DATA
           PASSED BY GBUS HAS SIZE OF LEVEL B1-1.
2750    C = 1! CHAIN TYPE IS MAIN CHAIN W/O PRIORITY.
2760    D = B2 !VISIT RATIO IS THE VISIT RATIO THAT HIT B1 AND STORED IN B2.
2770    GOSUB 7810
        !LOOP MACRO FOR GBUS DATA
           SERVICE
2780    A9$(5)="TAKE CARE OF LEVEL 1 UP TO LEVEL "+STR$(B1-1)+" BROADCAST."
2790    IF
           K9(8)=1
              THEN
                 GOSUB 10940
2800    GOSUB 5390
        !TAKE CARE OF LEVEL 1 UP TO LEVEL B1-1 BROADCAST.""
2810    A9$(5)="OVERFLOW FROM LEVEL "+STR$(B1)+" BROADCAST."
2820    IF
           K9(8)=1
              THEN
                 GOSUB 10940
2830    GOSUB 5640
        !TAKE CARE OF POSSIBLE OVERFLOW FROM LEVEL 1 UP TO LEVEL B1-1!
2840 NEXT B1
2850 A9$(5)="STB TRANSACTION."
2860 IF
        K9(8)=1
           THEN
              GOSUB 10940

2870 REM STB TRANSACTION.

2880 K9(6) = 1 - C9(2,0) !VISIT RATIO IS THE WRITE-RATIO.
2890 K9(5) = 1!STARTS FROM LEVEL 1.
2900 GOSUB 7470
     !COMPUTE LEVEL 1 FACILITY INDICATORS.
```

```
2910  A = 1!TYPE OF SERVICE FACILITY IS PE.
2920  B = 1! FOR LEVEL ONE.
2930  C = 1! CHAIN TYPE IS MAIN CHAIN W/O PRIORITY.
2940  D = K9(6) !VISIT RATIO IS THE WRITE-RATIO.
2950  B3 = S9(1,1) !STORE PE1 SERVICE TIME.
2960  S9(1,1) = S9(2,1) !LM SERVICE TIME.
2970  GOSUB 7600
      !LOOP MACRO FOR NON-GBUS FACILITIES.
2980  S9(1,1) = B3 !RESTORE PE1 SERVICE TIME.
2990  FOR B1 = 1 TO C9(0,0)-1!LEVELS THAT DO STB.
3000  |  GOSUB 5880
      |  !COMPUTE INCOMING/OUTGOING VISIT-RATIOS FOR STB.
3010  NEXT B1
3020  A9$(5)="ACK TRANSACTION."
3030  IF
         K9(8)=1
            THEN
               GOSUB 10940

3040  REM ACK TRANSACTIONS.

3050  K9(6) = 1 - C9(2,0) !VISIT RATIO IS WRITE-RATIO.
3060  FOR B1 = 2 TO C9(0,0) !ACKNOWLEDGE STARTS FROM LEVEL TWO.

3070  |  REM ACKNOWLEDGEMENT GENERATED BY LEVEL B1.

3080  |  IF
      |     B1=2
      |        THEN
      |           GOTO 3110
      |           !LEVEL 2 NEEDS TO ACKNOWLEDGE LEVEL 1 ONLY.
3090  |  K9(5) = B1-1!ACKNOWLEDGE 2 LEVEL ABOVE.
3100  |  GOSUB 6160
      |  !COMPUTE ACKNOWLEDGEMENT LOAD FOR A LEVEL.
3110  |  K9(5) = B1 !ACKNOWLEDGE ONE LEVEL ABOVE.
3120  |  GOSUB 6160
      |  !COMPUTE ACKNOWLEDGEMENT LOAD FOR A LEVEL.
3130  NEXT B1
3140  RETURN

3150  REM PERFORMANCE MEASURE COMPUTATION ROUTINE

3160  K9(0) = 0
3170  FOR B1 = 1 TO C9(3,0)
3180  |  B3 = 0
3190  |  FOR B2 = 1 TO 2
3200  |  |  B3 = B3 + V9(B2,B1)
3210  |  NEXT B2
3220  |  V9(4,B1) = B3 !TOTAL VS OF S.F. B.
3230  NEXT B1
3240  ON
         FNC1(K9(19))
            GOSUB 3360,3490
         ELSE
```

```
          GOTO 11460
          !COMPUTE OPEN/CLOSED SYSTEM THRUPUT AND RES. TIME.
3250  IF
        FNC3(K9(19))=1
          THEN
            GOTO 3260
          ELSE
            RETURN
3260  K9(18) = 2
3270  GOSUB 4220
      !PRINT OUT VS VALUES.
3280  FOR B1 = 1 TO C9(3,0)
3290  |  FOR B2 = 0 TO 4
3300  |  |  V9(B2,B1) = V9(B2,B1)*K9(1) !GET UTILIZATIONS
3310  |  NEXT B2
3320  NEXT B1
3330  ON
        FNC1(K9(19))
          GOSUB 3430,3530
        ELSE
          GOTO 11460
          !DISTINGUISH OPEN/CLOSED SYSTEM.
3340  RETURN

3350  REM COMPUTE OPEN SYSTEM THRUPUT AND RES. TIME.

3360  A = 4
3370  GOSUB 11220
      !FIND MAX VS PRODUCT.
3380  K9(1)=K9(13)/S2 !COMPUTE MAX OPEN SYSTEM THROUGHPUT.
3390  FOR B1 = 1 TO C9(3,0)
3400  |  K9(0) = K9(0) + V9(1,B1)/(1-K9(1)*V9(4,B1))
3410  NEXT B1
3420  RETURN
3430  FOR B1 = 3 TO 6
3440  |  K9(18) = B1 !TYPE OF PRINTOUT.
3450  |  GOSUB 4220
      |  !PRINT OUT TYPE K9(18) DATA.
3460  NEXT B1
3470  RETURN

3480  REM COMPUTE CLOSED CHAIN THRUPUT AND RES. TIME.

3490  GOSUB 4710
      !COMPUTE CLOSED CHAIN THRUPUT.
3500  GOSUB 11310
      !COMPUTE INFLATED CLOSED CHAIN NBAR FOR EVERY Q.
3510  GOSUB 11410
      !COMPUTE INFLATED CHAIN RES. TIME.
3520  RETURN
3530  RETURN !COMPUTE CLOSED SYSTEM PERFORMANCE.

3540  REM ERROR HANDLING ROUINTE
3550  IF
```

```
            ERR=22
               THEN
                  GOTO 3560
               ELSE
                  GOTO 4190
3560  PRINT ERR$(ERR)
3570  GOTO 1450
3580  IF
         ERR=22
            THEN
               GOTO 3590
            ELSE
               GOTO 4190
3590  PRINT ERR$(ERR)
3600  GOTO 1520
3610  IF
         ERR=22
            THEN
               GOTO 3620
            ELSE
               GOTO 4190
3620  PRINT ERR$(ERR)
3630  GOTO 1580
3640  IF
         ERR=22
            THEN
               GOTO 3650
            ELSE
               GOTO 4190
3650  PRINT ERR$(ERR)
3660  GOTO 4240
3670  IF
         ERR=22
            THEN
               GOTO 3680
            ELSE
               GOTO 4190
3680  PRINT ERR$(ERR)
3690  GOTO 4560
3700  IF
         ERR=22
            THEN
               GOTO 3710
            ELSE
               GOTO 4190
3710  PRINT ERR$(ERR)
3720  GOTO 4620
3730  IF
         ERR=22
            THEN
               GOTO 3740
            ELSE
               GOTO 4190
3740  PRINT ERR$(ERR)
```

```
3750  GOTO 6370
3760  IF
         ERR=22
            THEN
               GOTO 3770
            ELSE
               GOTO 4190
3770  PRINT ERR$(ERR)
3780  GOTO 6430
3790  IF
         ERR=22
            THEN
               GOTO 3800
            ELSE
               GOTO 4190
3800  PRINT ERR$(ERR)
3810  GOTO 6560
3820  IF
         ERR=22
            THEN
               GOTO 3830
            ELSE
               GOTO 4190
3830  PRINT ERR$(ERR)
3840  GOTO 6620
3850  IF
         ERR=22
            THEN
               GOTO 3860
            ELSE
               GOTO 4190
3860  PRINT ERR$(ERR)
3870  GOTO 6700
3880  IF
         ERR=22
            THEN
               GOTO 3890
            ELSE
               GOTO 4190
3890  PRINT ERR$(ERR)
3900  GOTO 6750
3910  IF
         ERR=22
            THEN
               GOTO 3920
            ELSE
               GOTO 4190
3920  PRINT ERR$(ERR)
3930  GOTO 8970
3940  IF
         ERR=22 OR ERR=8
            THEN
               GOTO 3960
3950  IF
```

```
        ERR=14
          THEN
            GOTO 3980
          ELSE
            GOTO 4190
3960  PRINT ERR$(ERR)
3970  GOTO 9000
3980  PRINT "NAME INCORRECT???"
3990  GOTO 8970
4000  IF
        ERR=22
          THEN
            GOTO 4010
          ELSE
            GOTO 4190
4010  PRINT ERR$(ERR)
4020  GOTO 9190
4030  IF
        ERR=22 OR ERR=8
          THEN
            GOTO 4040
          ELSE
            GOTO 4190
4040  PRINT ERR$(ERR)
4050  GOTO 9230
4060  IF
        ERR=22
          THEN
            GOTO 4070
          ELSE
            GOTO 4190
4070  PRINT ERR$(ERR)
4080  GOTO 10440
4090  IF
        ERR=22
          THEN
            GOTO 4100
          ELSE
            GOTO 4190
4100  PRINT ERR$(ERR)
4110  GOTO 10530
4120  IF
        ERR=22
          THEN
            GOTO 4130
          ELSE
            GOTO 4190
4130  PRINT ERR$(ERR)
4140  GOTO 10570
4150  IF
        ERR=22
          THEN
            GOTO 4160
          ELSE
```

```
                GOTO 4190
4160  PRINT ERR$(ERR)
4170  GOTO 10630

4180  REM OTHER ERRORS

4190  PRINT ERR$(ERL);" AT LINE ";ERL;", PLEASE RESTART TAD."
4200  GOTO 11460

4210  REM DRIVER TO PRINT ALL LEVELS

4220  ON
         ERROR
            GOTO 3640
4230  PRINT LIN(2)
4240  INPUT "ADJUST PAPER IF NECESSARY; TYPE YES WHEN READY! ";A9$(1)
4250  GOSUB 10180
      !CONVERT INPUT TO Y OR N OR NO-CHANGE.
4260  IF
         A9$(1) <> "Y"
            THEN
               GOTO 4240
4270  PRINT LIN(2)
4280  FOR P1 = 0 TO C9(0,0) !PRINT LEVEL 0 TO LEVEL MAX.
4290  |  GOSUB 5200
      |  !PRINT A MODEL LEVEL WITH DATA
4300  NEXT P1
4310  PRINT LIN(2)
4320  PRINT SPA(5);"FIG-";STR$(K9(18));": ";A7$(K9(18),1)
4330  PRINT
         SPA(5);"------ ";LEFT(
            "-----------------------------------------------------------------
            --",LEN(A7$(K9(18),1)))
4340  PRINT SPA(12);A7$(K9(18),2)
4350  PRINT
         SPA(12);LEFT(
            "-----------------------------------------------------------------
            --",LEN(A7$(K9(18),2)))
4360  RETURN

4370  REM SELECT THE COMBINATION OF POLICIES

4380  PRINT
         "********************************************************************
            **********"
4390  PRINT "YOU CAN SELECT THE COMBINATION OF POLICIES"
4400  PRINT "BY ENTERING THE SUM OF THE POLICY NUMBERS BELOW:"
4410  PRINT LIN(1)
4420  FOR S1 = 1 TO 5
4430  |  FOR S2 = 1 TO 2
4440  |  |  PRINT V8(S1,S2);" ";A8$(S1,S2);";",
4450  |  NEXT S2
4460  |  IF
      |     S1=4
```

```
               THEN
                  GOTO 4470
               ELSE
                  PRINT LIN(0)
4470  NEXT S1
4480  PRINT LIN(1)
4490  PRINT
         "***********************************************************************
            *********"
4500  PRINT "THE CURRENT COMBINATION OF POLICIES IS ";K9(19);" :"
4510  PRINT
         A8$(1,FNC1(K9(19)));", ";A8$(2,FNP1(K9(19)));", ";A8$(3,FNR2(K9(19)))
            ;
4520  PRINT ", ";A8$(4,FNC3(K9(19)));", AND ";A8$(5,FNP4(K9(19)));"."
4530  PRINT
         "***********************************************************************
            *********"
4540  ON
         ERROR
            GOTO 3670
4550  PRINT LIN(1)
4560  INPUT "IS THIS WHAT YOU WANT? CONFIRM YES/NO: ";A9$(1)
4570  GOSUB 10180
      !CONVERT INPUT TO Y OR N OR NO CHANGE
4580  IF
         A9$(1)<>"Y" AND A9$(1)<>"N"
            THEN
               GOTO 4560
4590  IF
         A9$(1)="Y"
            THEN
               RETURN
4600  ON
         ERROR
            GOTO 3700
4610  PRINT LIN(1)
4620  INPUT "ENTER THE SUM OF THE COMBINATION OF POLICIES! ";P2
4630  GOSUB 10230
      !CHECK NUMBER VALID
4640  ON
         S1
            GOTO 4650,4670,4690
         ELSE
            GOTO 11460
4650  K9(19)=P2 !VALID COMBINATION.
4660  GOTO 4380
4670  PRINT "THIS COMBINATION WILL BE IMPLEMENTED SOON!"
4680  GOTO 4380
4690  PRINT "INVALID COMBINATION!"
4700  GOTO 4380
4710  A = 2! FOR TYPE 2 CHAIN(THE UNBALANCED CHAIN)
4720  GOSUB 11220
      !GET THE MAX VS PRODUCT.
4730  P1 = S1 !INDEX FOR THE MAX VS PRODUCT.
```

```
4740  P2 = S2 !VALUE OF THE MAX VS PRODUCT.
4750  IF
         P2=0
            THEN
               STOP
            ELSE
               P2=1/P2 !MAX THROUGHPUT
4760  PRINT "MAX UNBALANCED CHAIN THROUGHPUT: ",P2
4770  K9(1)= 0!INITIALLY CLOSED CHAIN THROUGHPUT = 0!
4780  GOSUB 11030
      !INFLATE THE CLOSED CHAIN VS PRODUCT.
4790  GOSUB 11080
      !COMPUTE THE INFLATED CHAIN THROUGHPUT.
4800  P3 = K9(1) !SET BOUND
4810  P4 = K9(12)
4820  IF
         K9(12)<P2
            THEN
               GOTO 5000
            ELSE
               IF
                  V9(1,P1)>0
                     THEN
                        GOTO 4930
                        !CASE 1 AND 2!
4830  K9(1)= P2
4840  GOSUB 11030
4850  PRINT
         "V9(1,";P1;
            ") IS ZERO, SET THE UNBALANCED CHAIN FLOW TO MAX THROUGHPUT  =>"
4860  GOSUB 11080
4870  IF
         K9(12)< = P2
            THEN
               GOTO 4910
               !CASE 3.
4880  PRINT
         "CLOSED THROUGHPUT AT MAX UNBALANCED THROUGHPUT > MAX UNBALANCED THRO
            UGHPUT, SO NO SOLUTION."
4890  STOP !CASE 4

4900  REM CASE 3.

4910  PRINT
         "CLOSED THROUGHPUT AT MAX UNBALANCED THROUGHPUT <=  MAX UNBALANCED TH
            ROUGHPUT, SO THE SOLUTION EXISTS"
4920  GOTO 4970
4930  PRINT "CLOSED THROUGHPUT > MAX UNBALANCED THROUGHPUT";
4940  PRINT " BUT V9(1,";P1;") EQUALS TO ";
4950  PRINT V9(1,P1);" (>0) FOR THE CLOSED CHAIN,";
4960  PRINT " => THE SOLUTION EXISTS."
4970  K9(1)= P2 * .5
4980  PRINT LIN(1)
4990  GOTO 5020
```

```
5000  PRINT LIN(1) !CASE 1.
5010  K9(1)= K9(12)*.5SET INITIAL VALUE TO HALF CLOSED CHAIN THROUGHPUT.
5020  PRINT "THE UNBALANCED THROUGHPUT IS: ",K9(1)
5030  GOSUB 11030 !INFLATE.
5040  GOSUB 11080
      !COMPUTE THROUGHPUT.
5050  IF
         (ABS(K9(12) - K9(1)) / K9(1) ) < .001
            THEN
               RETURN !CONVERGES.
5060  IF
         K9(12)>P2
            THEN
               GOTO 5170
               !ESTIMATE > MAX THROUGHPUT.
5070  P5 = (K9(1)-K9(12))*(K9(1)-P4)/(P3-K9(12)+K9(1)-P4) !DIFFERENCE E.
5080  P3 = K9(12) !UPDATE BOUND
5090  P4 = K9(1) !UPDATE BOUND
5100  IF
         K9(12)>K9(1)
            THEN
               GOTO 5120
5110  IF
         K9(1)<=(K9(1)-P5) OR K9(12)>=(K9(1)-P5)
            THEN
               GOTO 5150
            ELSE
               GOTO 5130
5120  IF
         K9(12) <= (K9(1)- P5) OR K9(1) >= (K9(1)- P5)
            THEN
               GOTO 5150
5130  K9(1)= K9(1)- P5
5140  GOTO 5020
5150  K9(1)= (K9(12) + K9(1))/2
5160  GOTO 5020
5170  K9(1)= (P2 + K9(1))/2
5180  GOTO 5020

5190  REM PRINT A MODEL LEVEL WITH DATA.

5200  ON
         K9(18)
            GOSUB 7050,7180,7180,7180,7180,7280
         ELSE
            GOTO 11460
            !PREPARE DATA.
5210  GOSUB 8890
      !RESET MASK FOR A LEVEL.
5220  Q1 = 1
5230  GOSUB 9980
      !SET K8(0,1-5) WHICH INDICATE WHICH PART TO PRINT OUT.
5240  GOSUB 6800
      !PREPARE STRING FOR LINE(1-4)
```

```
5250  Q1 = 5
5260  GOSUB 9980
5270  GOSUB 6800
      !PREPARE STRINGS FOR LINE(5-8)0!
5280  FOR Q1 = 1 TO 8! PRINT LINE 1 TO 8 OF A LEVEL.
5290  |  GOSUB 9730
      |  !CONCATANATE AND PRINT.
5300  NEXT Q1
5310  IF
         P1>1
            THEN
               GOSUB 9730
5320  IF
         P1=1
            THEN
               PRINT "     |";SPA(7);"-------------"
5330  IF
         P1=0
            THEN
               PRINT SPA(31);"-------------"
5340  IF
         P1>0
            THEN
               PRINT "     |";SPA(28);"LEVEL ";STR$(P1)
5350  IF
         P1=0
            THEN
               PRINT SPA(37);"|" !LINE 10
5360  IF
         P1>0
            THEN
               PRINT "     |"
            ELSE
               PRINT SPA(37);"|" !LINE 11
5370  RETURN

5380  REM TAKE CARE OF LEVEL 1 UP TO LEVEL B1-1 BROADCAST OPERATION.

5390  FOR R1 = 1 TO B1-1!LEVEL 1 TO LEVEL B1-1!

5400  |  REM GC -> LBUS(DATA SIZE R1) -> PE -> LBUS(DATA SIZE R1) -> LSS.

5410  |  K9(5) = R1
5420  |  GOSUB 7470
      |  !COMPUTE FACILITY INDECIES.
5430  |  B = R1 !LEVEL IS R1.
5440  |  IF
      |     R1= 1
      |        THEN
      |           C=1
      |        ELSE
      |           C=2!LEVEL 1 IS THE MAIN CHAIN,OTHERS ARE UNBALANCED FLOW.
5450  |  D = B2 !VISIT RATIO IS THE HIT RATIO AT LEVEL B1.
5460  |  A = 3
```

```
5470  | GOSUB 7600
      | !-> GC
5480  | A = 0
5490  | GOSUB 7600
      | !-> LBUS
5500  | IF
      |    R1<>1
      |       THEN
      |          GOTO 5530
5510  | B3 = S9(1,1) !SAVE PE1 SERVICE TIME.
5520  | S9(1,1)=S9(2,1) !REPLACE BY LM SERVICE TIME.
5530  | A = 1
5540  | GOSUB 7600
      | !-> PE
5550  | IF
      |    R1=1
      |       THEN
      |          S9(1,1)=B3 !RESTORE PE1 SERVICE TIME.
5560  | IF
      |    R1=1
      |       THEN
      |          GOTO 5610
      |          !FOR LEVEL 1,NO LSS.
5570  | A = 0
5580  | GOSUB 7600
      | !-> LBUS
5590  | A = 2
5600  | GOSUB 7600
      | !-> LSS
5610  NEXT R1
5620  RETURN

5630  REM OVERFLOW TRANSACTION (VISIT-RATIO)*(SERVICE TIME) SUM COMPUTATION.

5640  FOR R1 = 1 TO B1-1!POSSIBLE OVERFLOW FROM A LEVEL B1 BROADCAST.
5650  | K9(5) = R1 !FOR LEVEL R1
5660  | GOSUB 7470
      | !COMPUTE FACILITY INDECIES.
5670  | B = R1 !-> LBUS(MSG) -> GC -> GBUS -> GC -> LBUS ->PE.
5680  | C = 2!OVERFLOW IS UNBALANCED FLOW.
5690  | D = B2*C9(4,R1) !VISIT RATIO IS B2*(PROBABILITY OF OVERFLOW LEVEL
      |    R1)
      |       O!
5700  | A = 0
5710  | GOSUB 7740
      | !-> LBUS
5720  | A = 3
5730  | GOSUB 7600
      | !-> GC
5740  | GOSUB 7880
      | !-> GBUS
5750  | K9(5) = R1 + 1
5760  | GOSUB 7470
5770  | A = 3
```

```
5780      B = K9(5)
5790      GOSUB 7600
          !-> GC
5800      A = 0
5810      GOSUB 7740
          !-> LBUS(MSG)
5820      A = 1
5830      GOSUB 7600
          !-> PE
5840   NEXT R1
5850   RETURN

5860   REM LBUS -> GC -> GBUS -> GC -> LBUS -> PE -> LBUS -> LSS.


5870   REM STB TRANSACTION VISIT RATIO COMPUTATION ROUTINE

5880   K9(5) = B1 !SET CURRENT LEVEL.
5890   GOSUB 7470
       !COMPUTE CURRENT LEVEL FACILITY INDICATORS.
5900   A=0!TYPE OF FACILITY IS LBUS.
5910   B=K9(5)
5920   C=2
5930   D=K9(6) !WRITE RATIO.
5940   GOSUB 7600 !LBUS
5950   A=3
5960   GOSUB 7600
       !-> GC
5970   GOSUB 7810
       !-> GBUS
5980   K9(5) = B1 + 1
5990   GOSUB 7470

6000   REM STB LBUS: DATA SIZE IS LAST LEVEL SIZE WHEN COMING IN.

6010   B = K9(5) !LEVEL IS B1+1!
6020   C = 2!UNBALANCED CHAIN W/O PRIORITY.
6030   D = K9(6)
6040   A = 3
6050   GOSUB 7600
       !-> GC
6060   A = 0
6070   GOSUB 7670
       !-> LBUS ;WITH DATA
          SIZE(B1-1)
6080   A = 1
6090   GOSUB 7600
       !-> PE
6100   A = 0
6110   GOSUB 7670
       !-> LBUS ;WITH DATA
          SIZE(B1-1)
6120   A = 2
6130   GOSUB 7600
```

```
     !-> LSS
6140 RETURN

6150 REM ACKNOWLEDGE A LEVEL: LBUS -> GC -> GBUS -> GC -> LBUS -> PE.

6160 GOSUB 7470
     !GIVEN A LEVEL IN K9(5)
6170 C = 2!UNBALANCED CHAIN W/O PRIORITY.
6180 D = K9(6) !ACK VISIT RATIO EQUALS TO WRITE RATIO.
6190 A = 0!LBUS
6200 B = K9(5)
6210 GOSUB 7740
     !LBUS MSG LOAD.
6220 A = 3!GC
6230 GOSUB 7600
     !GC SERVICE LOAD.
6240 GOSUB 7880
     !GBUS MSG LOAD.
6250 K9(5) = K9(5) - 1!FROM GBUS TO LAST LEVEL.
6260 GOSUB 7470
6270 A = 3!TYPE OF SERVICE FACILITY IS GC.
6280 B = K9(5)
6290 GOSUB 7600
     !GC SERVICE LOAD.
6300 A = 0!TYPE OF SERVICE FACILITY IS LBUS.
6310 GOSUB 7740
     !LBUS MSG LOAD.
6320 A = 1!FACILITY IS PE.
6330 GOSUB 7600
     !ADD PE LOAD.
6340 RETURN

6350 REM INPUT MODEL PARAMETERS FROM TERMINAL

6360 ON
        ERROR
           GOTO 3730
6370 INPUT "ENTER NUMBER OF LEVELS OF THE NEW MODEL: ";C9(0,0)
6380 IF
        C9(0,0)<=0 OR C9(0,0)-INT(C9(0,0))>0
           THEN
              PRINT A9$(6)
           ELSE
              IF
                 C9(0,0)>K9(17)
                    THEN
                       GOTO 6400
                    ELSE
                       GOTO 6420
6390 GOTO 6370
6400 PRINT "THE MAXIMUM NUMBER OF LEVELS IS ";K9(17);", PLEASE REENTER!"
6410 GOTO 6370
6420 ON
        ERROR
```

```
              GOTO 3760
6430  INPUT "ENTER NUMBER OF GBUS'S: ";C9(1,0)
6440  IF
         C9(1,0)>0 AND C9(1,0)-INT(C9(1,0))=0
            THEN
               GOTO 6460
            ELSE
               PRINT A9$(6)
6450  GOTO 6430
6460  PRINT LIN(1)
6470  PRINT "ENTER SERVICE TIMES IN NANO-SECONDS."
6480  PRINT LIN(1)
6490  GOSUB 10050
      !FEED A9$(1-4) WITH "LBUS,PE,LSS,GC"
6500  FOR R1 = 1 TO C9(0,0)
6510     FOR R2 = 0 TO 3
6520        A9$(5) = A9$(R2+1)+" SERVICE TIME AT LEVEL "+STR$(R1)+"? "
6530        IF
               R2=2 AND R1=1
                  THEN
                     A9$(5)="LOCAL MEMORY SERVICE TIME AT LEVEL 1? "
6540        PRINT A9$(5);
6550        ON
               ERROR
                  GOTO 3790
6560        INPUT S9(R2,R1)
6570        IF
               S9(R2,R1)>=0
                  THEN
                     GOTO 6590
                  ELSE
                     PRINT A9$(6)
6580        GOTO 6560
6590        A9$(5) = "NUMBER OF "+A9$(R2+1)+" AT LEVEL "+STR$(R1)+"? "
6600        PRINT A9$(5);
6610        ON
               ERROR
                  GOTO 3820
6620        INPUT C9(R2,R1)
6630        IF
               C9(R2,R1)>0 AND C9(R2,R1)-INT(C9(R2,R1))=0
                  THEN
                     GOTO 6650
                  ELSE
                     PRINT A9$(6)
6640        GOTO 6620
6650     NEXT R2
6660     C9(2,1)=0!NO LSS AT LEVEL 1 AND LOCAL MEMORY IS MERGED WITH PE.
6670     A9$(5) = "PROBABILITY OF OVERFLOW LEVEL "+STR$(R1)+"? "
6680     PRINT A9$(5);
6690     ON
            ERROR
               GOTO 3850
6700     INPUT C9(4,R1)
```

```
6710  |  IF
      |     C9(4,R1)>=0 AND C9(4,R1)<=1
      |        THEN
      |           GOTO 6730
      |        ELSE
      |           PRINT A9$(6)
6720  |  GOTO 6700
6730  NEXT R1
6740  ON
         ERROR
            GOTO 3880
6750  INPUT "GBUS/LBUS MESSAGE SERVICE TIME?";S9(0,0)
6760  IF
         S9(0,0)>=0
            THEN
               GOTO 6780
            ELSE
               PRINT A9$(6)
6770  GOTO 6750
6780  RETURN

6790  REM PREPARE STRINGS FOR PRINTING A LEVEL GIVEN LINE # INDIC. AND STRINGS

6800  FOR R1 = Q1 TO Q1+3! LINE(1,2,3,4) OR LINE(5,6,7,8)
6810  |  FOR R2 = 1 TO 5
6820  |  |  IF
      |  |     K8(0,R2) = 0
      |  |        THEN
      |  |           GOTO 6930
6830  |  |  IF
      |  |     K8(0,R2)=2
      |  |        THEN
      |  |           GOTO 6910
6840  |  |  R3 = R1 - INT(R1/5)*4 -1!(DATA AT 0,1,2,3 TH ROW AND COLUMN 1)
6850  |  |  S1 = F8(R3,R2) !GET NUMERICAL DATA
6860  |  |  S7$(0,0) = S8$(R3,(K9(18)-1)*5+R2) !S8$(,) IS PRESET AT SYSTEM
      |  |     INITIALIZATION.

6870  |  |  R4 = K8(1,R2)
6880  |  |  GOSUB 9800
      |  |  !SYNTHESIZE THE STRING.
6890  |  |  S7$(R1,R4) = S7$(0,0)
6900  |  |  GOTO 6930
6910  |  |  IF
      |  |     (P1=1 AND R1=1)
      |  |        THEN
      |  |           GOTO 6930
6920  |  |  S7$(R1,K8(1,R2))="            "
6930  |  NEXT R2
6940  NEXT R1
6950  IF
         P1>0
            THEN
               RETURN
```

```
6960  S7$(1,2) = "QQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQ"
6970  FOR R1 = 3 TO 7
6980  |  S7$(1,R1)=""
6990  NEXT R1
7000  S7$(2,4) = "      Q      "
7010  S7$(3,4) = "      Q      "
7020  S7$(4,4) = "-------------"
7030  RETURN

7040  REM PREPARE DATA FOR PRINT-OUT

7050  FOR R1 = 1 TO 5!GBUS, GC, PE, LSS, LBUS
7060  |  F8(0,R1) = -1!0 TH ROW BLANK.
7070  |  F8(3,R1) = -1!3RD ROW BLANK.
7080  |  IF
      |     R1=1
      |        THEN
      |           GOTO 7130
      |           !GBUS CASE.
7090  |  IF
      |     P1=0
      |        THEN
      |           R2=1
      |        ELSE
      |           R2=P1 !RESET LEVEL 0 TO LEVEL 1.
7100  |  F8(1,R1) = C9(K8(2,R1),R2) !# OF S.F.  'S.
7110  |  F8(2,R1) = S9(K8(2,R1),R2) !SERVICE TIME.
7120  |  GOTO 7150
7130  |  F8(1,R1) = C9(1,0) !FOR GBUS  ' 1ST ROW.
7140  |  F8(2,R1) = -1! FOR GBUS  ' 2ND ROW.
7150  NEXT R1
7160  RETURN

7170  REM CASE K9(18) = 2,3,4, AND 5.

7180  FOR R1 = 2 TO 5!GC, PE,LSS, LBUS.
7190  |  F8(0,R1) = -1!CURRENTLY NO PRIORITY.
7200  |  F8(3,R1) = -1!CURRENTLY NO LOW PRIORITY.
7210  |  IF
      |     P1=0
      |        THEN
      |           R2=1
      |        ELSE
      |           R2=P1 !RESET LEVEL 0 TO LEVEL 1.
7220  |  K9(5) = R2 !CURRENT LEVEL.
7230  |  GOSUB 7470
      |  !COMPUTE S.F. INDICATORS.
7240  |  ON
      |     K9(18)-1
      |        GOSUB 10690,10690,10750,10750
      |     ELSE
      |        GOTO 11460
7250  NEXT R1
7260  RETURN
```

```
7270  REM CASE 6(Q STATISTICS)

7280  FOR R1 = 2 TO 5!GC, PE, LSS, LBUS.
7290  |  IF
      |     P1=0
      |        THEN
      |           R2=1
      |        ELSE
      |           R2=P1 !RESET LEVEL 0 TO LEVEL 1.
7300  |  K9(5) = R2 !CURRENT LEVEL.
7310  |  GOSUB 7470
      |  !COMPUTE S.F. INDICATORS.
7320  |  R3 = V9(4,F9(K8(2,R1),R2)) !QUEUE UTILIZATION.
7330  |  F8(0,R1) = R3
7340  |  F8(1,R1) = R3/(1-R3) !NBAR.
7350  |  GOSUB 10850
      |  !COMPUTE 99% BUFFER SIZE.
7360  |  F8(2,R1) = S2 !99 BUFFER SIZE.
7370  |  F8(3,R1) = F8(1,R1)/K9(1) !RESPONSE TIME.
7380  NEXT R1
7390  R3 = V9(4,F9(0,0)) !UTILIZATION OF GBUS.
7400  F8(0,1) = R3 !GBUS UTILIZATION.
7410  F8(1,1) = R3/(1-R3) !GBUS NBAR.
7420  GOSUB 10850
      !GET 99 BUFFER SIZE.
7430  F8(2,1) = S2 !STORE 99% BUFFER SIZE.
7440  F8(3,1) = F8(1,1)/K9(1) !GBUS RESPONSE TIME.
7450  RETURN

7460  REM SERVICE FACILITY POINTER

7470  F9(0,0) = 1!GBUS IS THE STARTING FACILITY.
7480  F9(3,0) = C9(1,0) + 1!INITIAL VALLUE FOR LOOPING.
7490  S3 = C9(3,0) !SAVE THE VALUE OF # OF SERVICE FACILITIES.
7500  C9(3,0) = 0!SET INITIAL VAUE FOR LOOPING.
7510  FOR S1 = 1 TO K9(5) !AGGREATE UP TO LEVEL K9(5)0!
7520  |  F9(0,S1) = F9(3,S1-1) + C9(3,S1-1)
      |     !GBUS,LBUS,PE,LSS,GC,LBUS,PE,LSS,
      |        0!0!0!
7530  |  FOR S2 = 1 TO 3!LOOP ACCORDING THE ABOVE ORDER.
7540  |  |  F9(S2,S1) = F9(S2-1,S1) + C9(S2-1,S1)
7550  |  NEXT S2
7560  NEXT S1
7570  C9(3,0) = S3 !RESTORE C9(3,0) VAUE.
7580  RETURN

7590  REM LOOP MACRO FOR NON-GBUS SERVICE FACILITIES

7600  S2 = D*S9(A,B)/C9(A,B)
7610  FOR S1 = F9(A,B) TO F9(A,B)+C9(A,B)-1
7620  |  V9(C,S1) = V9(C,S1) + S2
7630  NEXT S1
7640  IF
```

```
        K9(8)=1
           THEN
              PRINT USING A9$(7),C9(A,B),A9$(8+A),B,D,S9(A,B),S2,C
7650  RETURN

7660  REM LOOP MACRO FOR STB-LBUS WHERE DATA SIZE IS FROM LAST LEVEL.

7670  S2 = D*S9(A,B-1)/C9(A,B)
7680  FOR S1 = F9(A,B) TO F9(A,B)+C9(A,B)-1
7690  |  V9(C,S1) = V9(C,S1) + S2
7700  NEXT S1
7710  IF
        K9(8)=1
           THEN
              PRINT USING A9$(7),C9(A,B),A9$(8+A),B,D,S9(A,B-1),S2,C
7720  RETURN

7730  REM LOOP MACRO FOR LBUS MSG LOAD COMPUTATION

7740  S2 = D*S9(0,0)/C9(A,B)
7750  FOR S1 = F9(A,B) TO F9(A,B)+C9(A,B)-1
7760  |  V9(C,S1) = V9(C,S1) + S2
7770  NEXT S1
7780  IF
        K9(8)=1
           THEN
              PRINT USING A9$(7),C9(A,B),A9$(8+A),B,D,S9(0,0),S2,C
7790  RETURN

7800  REM LOOP MACRO FOR GBUS DATA LOAD COMPUTATION

7810  S2 = D*S9(0,B)/C9(1,0)
7820  FOR S1 = F9(0,0) TO F9(0,0)+C9(1,0)-1
7830  |  V9(C,S1) = V9(C,S1) + S2
7840  NEXT S1
7850  IF
        K9(8)=1
           THEN
              PRINT USING A9$(7),C9(1,0),"GBUS",B,D,S9(0,B),S2,C
7860  RETURN

7870  REM LOOP MACRO FOR GBUS MSG LOAD COMPUTATION

7880  S2 = D*S9(0,0)/C9(1,0)
7890  FOR S1 = F9(0,0) TO F9(0,0)+C9(1,0)-1
7900  |  V9(C,S1) = V9(C,S1) + S2
7910  NEXT S1
7920  IF
        K9(8)=1
           THEN
              PRINT USING A9$(7),C9(1,0),"GBUS",B,D,S9(0,0),S2,C
7930  RETURN
7940  REM INITIALIZE TEXT
```

```
7950  FOR S1 = 0 TO 3
7960  |   FOR S2= 1 TO 30
7970  |   |   READ S8$(S1,S2)
7980  |   NEXT S2
7990  NEXT S1
8000  DATA
          '',
          '',
          '',
          '',
          '',
          '',
          '',
          '',
          '',
          '',
          '',
          '',
          '',
          '',
          '',
          '',
          '',
          '',
          '',
          ''
8010  DATA
          '',
          '',
          '',
          '',
          '',
          'U',
          'U',
          'U',
          'U',
          'U'
8020  DATA
          'GBUS',
          'GC',
          'PE',
          'LSS',
          'LBUS',
          'V1',
          'V1',
          'V1',
          'V1',
          'V1',
          'U1',
          'U1'
8030  DATA
          'U1',
          'U1',
          'U1',
```

```
          'N1',
          'N1',
          'N1',
          'N1',
          'N1',
          'R1',
          'R1',
          'R1',
          'R1',
          'R1'
8040 DATA
          'N',
          'N',
          'N',
          'N',
          'N'
8050 DATA
          '',
          'ns',
          'ns',
          'ns',
          'ns',
          'V2',
          'V2',
          'V2',
          'V2',
          'V2',
          'U2',
          'U2',
          'U2',
          'U2'
8060 DATA
          'U2',
          'N2',
          'N2',
          'N2',
          'N2',
          'N2',
          'R2',
          'R2',
          'R2',
          'R2',
          'R2',
          'B',
          'B',
          'B',
          'B',
          'B'
8070 DATA
          '',
          '',
          '',
          '',
          '',
```

```
     '',
     '',
     '',
     '',
     '',
     '',
     '',
     '',
     '',
     '',
     '',
     '',
     '',
     '',
     ''
8080 DATA
     '',
     '',
     '',
     '',
     '',
     'R',
     'R',
     'R',
     'R',
     'R'
8090 K9(19) = 11111
8100 A9$(6) = "INVALID INPUT, PLEASE REENTER"
8110 A9$(7)=
     "######   >########## ##### #####.##### ########.### ########.# #####
     #"
8120 A9$(8)="LBUS"
8130 A9$(9)="PE"
8140 A9$(10)="LSS"
8150 A9$(11)="GC"
8160 FOR S1 = 1 TO 8!INITIALIZE TABLES FOR MAPPING DATA AND INDICATORS.
8170 |  FOR S2 = 1 TO 5
8180 |  |  READ K8(S1,S2)
8190 |  NEXT S2
8200 NEXT S1
8210 DATA
     1,
     2,
     4,
     6,
     7
8220 DATA
     1,
     3,
     1,
     2,
     0
8230 DATA
     2,
```

```
          2,
          2,
          2,
          2
8240  DATA
          0,
          0,
          2,
          2,
          1
8250  DATA
          0,
          0,
          0,
          0,
          1
8260  DATA
          1,
          2,
          1,
          2,
          2
8270  DATA
          0,
          1,
          2,
          2,
          2
8280  DATA
          0,
          1,
          1,
          1,
          0

8290  REM INITIALIZE LEVEL FORMAT

8300  FOR S1 = 1 TO 9
8310  |   FOR S2 = 1 TO 7
8320  |   |   READ S7$(S1,S2)
8330  |   |   S7$(S1+9,S2) = S7$(S1,S2)
8340  |   NEXT S2
8350  NEXT S1
8360  DATA
          '     |        ',
          '-------------',
          '-----'
8370  DATA
          '-------------',
          '-----',
          '-------------',
          ''
8380  DATA
          '     |        ',
```

```
                '      |      ',
                '     '
8390  DATA
                '      |      ',
                '     ',
                '      |      ',
                ''
8400  DATA
                '     |       ',
                '      |      ',
                '     '
8410  DATA
                '      |      ',
                '     ',
                '      |      ',
                ''
8420  DATA
                '     |       ',
                '-------------',
                '     '
8430  DATA
                '-------------',
                '     ',
                '-------------',
                ''
8440  DATA
                '     |       ',
                '',
                '     '
8450  DATA
                '',
                '     ',
                '',
                ''
8460  DATA
                '     |_______',
                '',
                '     '
8470  DATA
                '',
                '     ',
                '',
                ''
8480  DATA
                '     |       ',
                '|           |',
                '     '
8490  DATA
                '|           |',
                '     ',
                '|           |',
                ''
8500  DATA
                '     |       ',
```

```
           '',
           '    '
8510  DATA
           '',
           '    ',
           '',
           ''
8520  DATA
           '      |      ',
           '-------------',
           '    '
8530  DATA
           '-------------',
           '    ',
           '-------------',
           ''
8540  FOR S1 = 1 TO 5
8550  |  FOR S2 = 1 TO 2
8560  |  |  READ A8$(S1,S2)
8570  |  NEXT S2
8580  NEXT S1
8590  DATA
           "OPEN",
           "CLOSED"
8600  DATA
           "PERCOLATE",
           "PARALLEL"
8610  DATA
           "RETRANSMIT",
           "RESERVE SPACE"
8620  DATA
           "A (LOCALITY,READ%) POINT",
           "A LOCALITY SET GIVEN A READ%"
8630  DATA
           "EQUAL PRIORITY",
           "STB LOW PRIORITY"
8640  FOR S1 = 1 TO 5
8650  |  FOR S2 = 1 TO 2
8660  |  |  READ V8(S1,S2)
8670  |  NEXT S2
8680  NEXT S1
8690  DATA
           10000,
           20000
8700  DATA
           1000,
           2000
8710  DATA
           100,
           200
8720  DATA
           10,
           20
8730  DATA
```

```
          1,
          2
8740  FOR S1 = 1 TO 6
8750  |   FOR S2 = 1 TO 2
8760  |   |   READ A7$(S1,S2)
8770  |   NEXT S2
8780  NEXT S1
8790  DATA
          "NUMBER OF SERVICE FACILITIES AND THEIR SERVICE TIMES.",
          ""
8800  DATA
          "SUM OF (VISIT RATIO)*(SERVICE TIME) -- 1(MAIN CHAIN),"
8810  DATA
          "2(UAP CHAIN)"
8820  DATA
          "UTILIZATIONS -- 1(MAIN CHAIN), 2(UAP CHAIN).",
          ""
8830  DATA
          "MEAN QUEUE LENGTH -- 1(MAIN CHAIN), 2(UAP).",
          ""
8840  DATA
          "RESPONSE TIME -- 1(MAIN CHAIN), 2(UAP CHAIN).",
          ""
8850  DATA
          "FACILITY MEASURES -- U(UTILIZATION), N(MEAN QUEUE LENGTH),"
8860  DATA
          "B(99% PROBABILITY BUFFER SIZE), AND R(RESPONSE TIME)."
8870  RETURN

8880  REM RESTORE THE LEVEL FORMAT

8890  FOR S1 = 1 TO 9
8900  |   FOR S2 = 1 TO 7
8910  |   |   S7$(S1,S2) = S7$(S1+9,S2)
8920  |   NEXT S2
8930  NEXT S1
8940  RETURN

8950  REM READ MODEL PARAMETERS FROM SAVED FILE A9$(0)

8960  ON
          ERROR
              GOTO 3910
8970  INPUT "ENTER THE OLD MODEL'S NAME! ";A9$(0)
8980  DEFINE FILE #1=A9$(0)
8990  ON
          ERROR
              GOTO 3940
9000  READ #1,C9(0,0) !READ NUMBER OF LEVELS FIRST.
9010  READ #1,C9(1,0) !READ NUMBER OF GBUS  'S IN THE MODEL.
9020  FOR S1= 1 TO C9(0,0)
9030  |   FOR S2 = 0 TO 3
9040  |   |   READ #1,C9(S2,S1)
9050  |   NEXT S2
```

```
9060  |  READ #1,C9(4,S1)
9070  |  IF
      |     C9(4,S1)>1 OR C9(4,S1)<0
      |        THEN
      |           GOTO 9080
      |        ELSE
      |           GOTO 9100
9080  |  PRINT "INVALID PROBABILITY AT LEVEL ";S1
9090  |  GOTO 11460
9100  |  FOR S2 = 0 TO 3
9110  |  |  READ #1,S9(S2,S1)
9120  |  NEXT S2
9130  NEXT S1
9140  READ #1,S9(0,0)
9150  C9(2,1)=0!NO LSS AT LEVEL 1 AND LOCAL MEMORY IS MERGED WITH PE.
9160  RETURN

9170  REM SAVE MODEL PARAMETERS

9180  ON
         ERROR
            GOTO 4000
9190  INPUT "ENTER A NAME TO SAVE THE MODEL! ";A9$(0)
9200  DEFINE FILE #1=A9$(0)
9210  GOSUB 10050
      !SET A9$(1-4) TO "LBUS,PE,LSS,GC"
9220  ON
         ERROR
            GOTO 4030
9230  WRITE #1,C9(0,0)," , NUMBER OF LEVELS OF THE MODEL."
9240  WRITE #1,C9(1,0)," , NUMBER OF GBUS IN THE MODEL."
9250  FOR S1= 1 TO C9(0,0)
9260  |  FOR S2 = 0 TO 3
9270  |  |  WRITE
      |  |     #1,C9(S2,S1)," , NUMBER OF "+A9$(S2+1)+" AT LEVEL "+STR$(S1)+
      |  |        "."
9280  |  NEXT S2
9290  |  WRITE #1,C9(4,S1)," , PROBABILITY OF OVERFLOW LEVEL "+STR$(S1)+"."
9300  |  WRITE #1,S9(0,S1)," , LBUS DATA SERVICE TIME AT LEVEL "+STR$(S1)+"."
9310  |  FOR S2 = 1 TO 3
9320  |  |  WRITE
      |  |     #1,S9(S2,S1)," , "+A9$(S2+1)+" SERVICE TIME AT LEVEL "+STR$(S1)
      |  |        +"."
9330  |  NEXT S2
9340  NEXT S1
9350  WRITE #1,S9(0,0)," , "+"GBUS/LBUS MESSAGE SERVICE TIME."
9360  RETURN

9370  REM COMPUTE NUMBER OF SERVICE FACILITIES.

9380  S3 = C9(1,0)
9390  FOR S1 = 1 TO C9(0,0)
9400  |  FOR S2 = 0 TO 3
9410  |  |  F9(S2,S1) = 0
```

```
9420  |  |  S3 = S3 + C9(S2,S1)
9430  |  NEXT S2
9440  NEXT S1
9450  IF
         S3 > K9(16)
            THEN
               GOTO 9530
9460  C9(3,0) = S3
9470  FOR S1 = 0 TO 4!INITIALIZE VISIT-RATIO AND PERFORMANCE BUFFERS
9480  |  FOR S2 = 1 TO C9(3,0)
9490  |  |  V9(S1,S2) = 0
9500  |  NEXT S2
9510  NEXT S1
9520  RETURN
9530  PRINT "TOO MANY SERVICE FACILITIES IN THE MODEL("+STR$(S3)+")!"
9540  PRINT LIN(1)
9550  PRINT "REDUCE MODEL SIZE OR CALL RICH WANG FOR HELP."
9560  GOTO 11460

9570  REM SYSTEM RESET FOR A GIVEN SET OF (READ %, LOACALITY)

9580  IF
         FNC3(K9(19))=1
            THEN
               S3=C9(5,1)
            ELSE
               S3=C9(5,1)+.1
9590  FOR S1 = 1 TO C9(0,0)
9600  |  C9(5,S1) = S3 !SET LOCALITIES FOR ALL THE LEVELS
9610  |  FOR S2 = 0 TO 3
9620  |  |  F9(S2,S1) = 0! RESET THE FACILITY INDICATOR
9630  |  NEXT S2
9640  NEXT S1
9650  C9(5,C9(0,0)) = 1!LOCALITY AT THE FLOOR IS 1
9660  FOR S1 = 0 TO 4! CLEAR VISIT RATIO AND PERFORMANCE BUFFERS
9670  |  FOR S2 = 1 TO C9(3,0)
9680  |  |  V9(S1,S2) = 0
9690  |  NEXT S2
9700  NEXT S1
9710  RETURN

9720  REM CONCATANATE AND PRINT A LINE

9730  S7$(0,0) = ""
9740  FOR S1 = 1 TO 7
9750  |  S7$(0,0) = S7$(0,0)+S7$(Q1,S1)
9760  NEXT S1
9770  PRINT S7$(0,0)
9780  RETURN

9790  REM FORMAT A LINE SEGMENT GIVEN [S1,S7$(0,0),R4]
9800  IF
         S1 < 0
            THEN
```

```
                GOTO 9920
9810  A9$(3) = LEFT(STR$(S1),8)
9820  S2 = 12 - LEN(A9$(3)) - LEN(S7$(0,0))
9830  S3 = INT(S2/2)
9840  S2 = S2 - S3
9850  A9$(1) = "        "
9860  A9$(2) = "|"
9870  IF
         R4 = 1 OR R4 = 7
            THEN
               GOTO 9880
            ELSE
               GOTO 9890
9880  A9$(2) = " "
9890  S7$(0,0) = LEFT(A9$(2)+A9$(1),S2)+A9$(3)+" "+S7$(0,0)
9900  S7$(0,0) = S7$(0,0) + RIGHT(A9$(1)+A9$(2),8-S3)
9910  RETURN
9920  IF
         R4=1 OR R4=7
            THEN
               GOTO 9950
9930  S7$(0,0)="|           |"
9940  RETURN
9950  S7$(0,0)="             "
9960  RETURN

9970  REM SET LEVEL > 0 FOR PRINT OUT A LEVEL

9980  IF
         Q1 = 1
            THEN
               S2=3
            ELSE
               IF
                  Q1=5
                     THEN
                        S2=6
                     ELSE
                        GOTO 11460
9990  IF
         P1>1
            THEN
               S3=2
            ELSE
               S3=P1
10000 FOR S1=1 TO 5
10010 |  K8(0,S1) = K8(S3+S2,S1)
10020 NEXT S1
10030 RETURN

10040 REM SET A9$(1-4)
10050 A9$(1) = "LBUS"
10060 A9$(2) = "PE"
10070 A9$(3) = "LSS"
```

```
10080  A9$(4) = "GC"
10090  RETURN

10100  REM PRINT OUT SYS.THRUPUT/RES.

10110  PRINT LIN(1)
10120  PRINT "(LOCALITY,READ%)=(";STR$(C9(5,1));",";STR$(C9(2,0));"), ";
10130  PRINT "=> (SYSTEM-THROUGHPUT,SYSTEM RESPONSE TIME)=(";
10140  PRINT K9(1);",";K9(0);")."
10150  WRITE #2,K9(19),C9(2,0),C9(5,1),K9(0),K9(1) ! POLICY COMBINATION;
          READ%;
             LOCALITY; RES. TIME; THRUPUT.
10160  RETURN

10170  REM CONVERT INPUT TO Y OR N

10180  A9$(1)=CVT$$(A9$(1),32)
10190  IF
          A9$(1)="Y" OR A9$(1)="YES"
             THEN
                A9$(1)="Y"
10200  IF
          A9$(1)="N" OR A9$(1)="NO"
             THEN
                A9$(1)="N"
10210  RETURN

10220  REM CHECK SUM VALID

10230  S1 = 1
10240  S2 = FNC1(P2)*10000
10250  IF
          S2 <>V8(1,1) AND S2<>V8(1,2)
             THEN
                GOTO 10380
10260  S2 = FNP1(P2)*1000
10270  IF
          S2 <>V8(2,1) AND S2<>V8(2,2)
             THEN
                GOTO 10380
10280  IF
          S2=V8(2,2)
             THEN
                S1=2
10290  S2 = FNR2(P2)*100
10300  IF
          S2<>V8(3,1) AND S2<>V8(3,2)
             THEN
                GOTO 10380
10310  IF
          S2=V8(3,2)
             THEN
                S1=2
10320  S2 = FNC3(P2)*10
```

```
10330  IF
          S2<>V8(4,1) AND S2<>V8(4,2)
             THEN
                GOTO 10380
10340  S2 = FNP4(P2)
10350  IF
          S2<>V8(5,1) AND S2<>V8(5,2)
             THEN
                GOTO 10380
10360  IF
          S2=V8(5,2)
             THEN
                S1=2
10370  RETURN
10380  S1 = 3!INVALID COMBINATION.
10390  RETURN

10400  REM SET UP PARAMETERS FOR POINT/CURVE ESTIMATES, OPEN/CLOSED SYSTEM.

10410  ON
          FNC3(K9(19))
             GOTO 10420,10490
          ELSE
             GOTO 11460
10420  ON
          ERROR
             GOTO 4060
10430  PRINT LIN(1)
10440  INPUT "ENTER A LOCALITY(ASSUME THE SAME ACROSS LEVELS)! ";C9(5,1)
10450  IF
          C9(5,1)>=0 AND C9(5,1)<=1
             THEN
                GOTO 10470
             ELSE
                PRINT A9$(6)
10460  GOTO 10440
10470  C9(5,0) = 1!COUNTER FOR LOCALITIES TO MEASURE IS SET TO 1
10480  GOTO 10510
10490  C9(5,0) = 9!SET COUNTER TO 9 TO GET AN INCREMENT OF 0.1
10500  C9(5,1) = 0!SO THAT THE FIRST LOCALITY IS 0.1
10510  ON
          ERROR
             GOTO 4090
10520  PRINT LIN(1)
10530  INPUT "ENTER READ%! ";C9(2,0)
10540  IF
          C9(2,0)>=0 AND C9(2,0)<=1
             THEN
                GOTO 10560
             ELSE
                PRINT A9$(6)
10550  GOTO 10530
10560  ON
          FNC1(K9(19))
```

```
            GOTO 10570,10630
         ELSE
            GOTO 11460
10570  ON
         ERROR
            GOTO 4120
10580  PRINT LIN(1)
10590  INPUT "MAXIMUM UTILITY(<1) ALLOWED FOR A SERVICE FACILITY? ";K9(13)
10600  IF
         K9(13)>0 AND K9(13)<1
            THEN
               RETURN
            ELSE
               PRINT A9$(6)
10610  GOTO 10590

10620  REM CLOSED SYSTEM

10630  ON
         ERROR
            GOTO 4150
10640  PRINT LIN(1)
10650  INPUT "ENTER THE POPULATION IN THE CLOSED CHAIN!";K9(14)
10660  IF
         K9(14)>0 AND K9(14)-INT(K9(14))=0 AND K9(14)<=K9(15)
            THEN
               RETURN
            ELSE
               PRINT A9$(6)
10670  GOTO 10650

10680  REM PRIMITIVES FOR PRINTOUT ROUTINE(82200):CASE 2 & 3.

10690  FOR S1 = 1 TO 2!FIRST AND 2ND ROW DATA
10700  |  F8(S1,R1) = V9(S1,F9(K8(2,R1),R2)) !VISIT RATIOS.
10710  |  F8(S1,1) = V9(S1,F9(0,0)) !FOR GBUS.
10720  NEXT S1
10730  RETURN

10740  REM PRINTOUT ROUTINE PRIMITIVES, CASE 4 & 5.

10750  FOR S1 = 1 TO 2!1ST ROW AND 2ND ROW.
10760  |  S2 = F9(K8(2,R1),R2)
10770  |  F8(S1,R1) = V9(S1,S2)/(1-V9(1,S2)-V9(2,S2)) !NBAR.
10780  |  F8(S1,1) = V9(S1,F9(0,0))/(1-V9(1,F9(0,0))-V9(2,F9(0,0))) !GBUS
10790  |  ON
       |     K9(18)-3
       |        GOTO 10820,10800
       |     ELSE
       |        GOTO 11460
10800  |  F8(S1,R1) = F8(S1,R1)/K9(1) !RESPONSE TIME.
10810  |  F8(S1,1) = F8(S1,1)/K9(1) !FOR GBUS.
10820  NEXT S1
10830  RETURN
```

```
10840  REM CALCULATE 99% BUFFER SIZE S2.

10850  S1 = 1-R3 !NOT USED
       IF
          NO CUSTOMER.
10860  S2 = 0!INITIALLY SIZE = 0!
10870  S3 = S1 !INITIAL PROBABILITY.
10880  IF
          S3>.99
             THEN
                RETURN !CUMULATIVE PROBABILITY EXCEEDS .99
10890  S1 = S1*R3 !NEXT QUEUE SIZE PROBABILITY.
10900  S3 = S3 + S1 !ACCUMULATE PROBABILITY.
10910  S2 = S2 +1
10920  IF
          S2=999
             THEN
                RETURN
             ELSE
                GOTO 10880

10930  REM VISIT RATIO REPORT HEADING

10940  PRINT LIN(1)
10950  PRINT A9$(5)
10960  PRINT
          LEFT("------------------------------------------------------------",
             LEN(A9$(5)))
10970  PRINT LIN(1)
10980  PRINT
          "NUMBER OF FACILITIES LEVEL VISIT-RATIO SERVICE-TIME VS-PRODUCT CHAIN
             -TYPE"
10990  PRINT
          "-------------------- ----- ----------- ----------- ---------- -----
             -----"
11000  PRINT LIN(1)
11010  RETURN

11020  REM INFLATE THE CLOSED CHAIN

11030  FOR S1 = 1 TO C9(3,0)
11040  |  IF
       |     V9(1,S1)=0
       |        THEN
       |           V9(5,S1)=0
       |        ELSE
       |           V9(5,S1)=V9(1,S1)/(1-V9(2,S1)*K9(1))
11050  NEXT S1
11060  RETURN
11070  REM BUZEN'S NC ALGORITHM

11080  FOR S1 = 1 TO K9(14) !POPULATION
11090  |  G(S1) = 0
```

```
11100  NEXT S1
11110  G(0) = 1
11120  FOR S1 = 1 TO C9(3,0) !# OF S.F. 'S
11130  |  FOR S2 = 1 TO K9(14) !POPULATION
11140  |  |  G(S2) = G(S2) + V9(5,S1)*G(S2 - 1)
11150  |  NEXT S2
11160  NEXT S1
11170  K9(12) = G(K9(14) - 1)/G(K9(14))
11180  PRINT "THE CLOSED CHAIN THROUGHPUT IS: ",K9(12)
11190  PRINT LIN(1)
11200  RETURN

11210  REM FIND THE MAX VS PRODUCT.

11220  S1 = 0
11230  S2 = 0
11240  FOR S3 = 1 TO C9(3,0)
11250  |  IF
       |     S2>=V9(A,S3)
       |        THEN
       |           GOTO 11280
11260  |  S1 = S3
11270  |  S2 = V9(A,S3)
11280  NEXT S3
11290  RETURN

11300  REM COMPUTE NBAR OF EACH QUEUE FROM BUZEN'S ALGORITHM.

11310  FOR S2 = 1 TO C9(3,0)
11320  |  V9(6,S2) = 0
11330  |  S3 = 1
11340  |  FOR S1 = 1 TO K9(14)
11350  |  |  S3 = S3 *V9(5,S2)
11360  |  |  V9(6,S2) = V9(6,S2) + S3*G(K9(14)-S1)/G(K9(14))
11370  |  NEXT S1
11380  NEXT S2
11390  RETURN

11400  REM COMPUTE CLOSED CHAIN RES. TIME.

11410  K9(0) = 0
11420  FOR S1 = 1 TO C9(3,0)
11430  |  K9(0) = K9(0) + V9(6,S1)/K9(1)
11440  NEXT S1
11450  RETURN
11460  STOP !IMPOSSIBLE CONDITION.
```

Appendix IV:

Listing of Simulation Program of P1L3 Model using RESQ

This program simulates the P1L3 model of the INFOPLEX data storage hierarchy. It uses the RESQ package which is available under the userid "RESCUE" on the IBM/370 at the Information Processing Service, Massachusetts Institute of Technology. Permission from Professor Stuart E. Madnick is required before using RESQ.

```
MODEL:TADP1L3  /* A RESQ P1L3 MODEL TO COMPARE WITH TAD */

METHOD:APLOMB /* SIMULATION METHOD IS USED */

/**************************************************/
/*                MODEL PARAMETERS                */
/**************************************************/

NUMERIC PARAMETERS: CPU_SEC /* CPU SECONDS */
NUMERIC PARAMETERS: HIGH /* HIGH PRIORITY */
NUMERIC PARAMETERS: LOW  /* LOW PRIORITY */
NUMERIC PARAMETERS: MAXMP /* MAXIMUM DEGREE OF MULTIPROGRAMMING */
NUMERIC PARAMETERS: MEDIUM /* MEDIUM PRIORITY */
NUMERIC PARAMETERS: PIN1  /* PROBABILITY THAT DATA IN LEVEL 1 */
NUMERIC PARAMETERS: PIN2  /* PROBABILITY THAT DATA IN LEVEL 2 */
NUMERIC PARAMETERS: PIN3  /* PROBABILITY THAT DATA IN LEVEL 3 */
NUMERIC PARAMETERS: POV1  /* PROBABILITY TO OVERFLOW LEVEL 1 */
NUMERIC PARAMETERS: POV2  /* PROBABILITY TO OVERFLOW LEVEL 2 */
NUMERIC PARAMETERS: PREAD /* PERCENTAGE OF READ TRANSACTION */
NUMERIC PARAMETERS: SIM_TIME /* SIMULATION TIME */

/**************************************************/
/*                MODEL IDENTIFIERS               */
/**************************************************/

NUMERIC IDENTIFIERS:BEXM  /* MESSAGE EXECUTION TIME AT BUS */
   BEXM:100               /* 100 NANO SECONDS */
NUMERIC IDENTIFIERS:BEXD1 /* DATA EXECUTION TIME AT LEVEL 1 BUS */
   BEXD1:100
NUMERIC IDENTIFIERS:BEXD2 /* BUS DATA EXECUTION TIME(LEVEL 2) */
   BEXD2:800
NUMERIC IDENTIFIERS:DEX1  /* DEVICE DATA EXECUTION TIME(LEVEL 1) */
   DEX1:100
NUMERIC IDENTIFIERS:DEX2  /* DEVICE DATA EXECUTION TIME(LEVEL 2) */
   DEX2:1000
NUMERIC IDENTIFIERS:DEX3  /* DEVICE DATA EXECUTION TIME(LEVEL 3) */
   DEX3:2000
NUMERIC IDENTIFIERS:INTARRTIME /* INTER ARRIVAL TIME */
   INTARRTIME:999999999
NUMERIC IDENTIFIERS:KEX /* CONTROLLER EXECUTION TIME */
   KEX:100
NUMERIC IDENTIFIERS:REX /* MEMORY REQUEST EXECUTION TIME */
   REX:200
NUMERIC IDENTIFIERS:ZERO /* ZERO SERVICE TIME */
   ZERO: 0

/**************************************************/
/*     SIMULATION TIME DEPENDENT VARIABLES        */
/**************************************************/

GLOBAL VARIABLES: CLOCK /* CURRENT SIMULATION CLOCK */
   CLOCK: 0             /* INITIALIZED TO ZERO */
GLOBAL VARIABLES: MRESP /* MEAN RESPONSE TIME */
   MRESP: 0             /* INITIALIZED TO ZERO */
```

```
GLOBAL VARIABLES: NTXN  /* ELAPSED TIME OF ALL TRANSACTIONS */
   NTXN: 0             /* INITIALIZED TO ZERO */
GLOBAL VARIABLES: SUMW  /* ELAPSED TIME OF ALL TRANSACTIONS */
   SUMW: 0             /* INITIALIZED TO ZERO */

/*********************************************************/
/*                                                       */
/* KEYS: D(DEVICE); G(GBUS); L(LBUS); K(CONTROLLER)      */
/*        M(MEMORY REQUEST PROCESSOR)                    */
/*       FM,FD(BUS_FACILITY TO PROCESS BEXM OR BEXD)     */
/* E.G.  FD1L1 = FACILITY LBUS1 PROCESSES BEXD1          */
/*********************************************************/

NODE ARRAYS: DX21(2) DX22(2)
NODE ARRAYS: FD1G(2) FD1L1(2) FD1L2(5)
NODE ARRAYS: FD2G(2) FD2L2(5) FD2L3(5)
NODE ARRAYS: FMG(6) FML1(3) FML2(9) FML3(4)
NODE ARRAYS: KI1(3) KI2(6) KI3(3)
NODE ARRAYS: KX1(2) KX2(4) KX3(2)
NODE ARRAYS: MI2(3) MI3(2)
NODE ARRAYS: MX2(2)

MAX JV:0 /* ONE JOB VARIABLE PER JOB */

/**************************************************/
/*            QUEUE DEFINITIONS                   */
/**************************************************/

QUEUE: START         /* COLLECT THROUGHPUT */
   TYPE: FCFS
   CLASS LIST:       STAR1
      SERVICE TIMES: ZERO*DISCRETE(1,1)

QUEUE:D1            /* LEVEL 1 DEVICE: CACHE */
   TYPE:PRTY
   CLASS LIST:       PRDI1R                PRDI1W
      SERVICE TIMES: REX*DISCRETE(1,1)     DEX1*DISCRETE(1,1)
      PRIORITIES:    HIGH                  HIGH

   CLASS LIST:       DI1R                  DX1
      SERVICE TIMES: DEX1*DISCRETE(1,1)    DEX1*DISCRETE(1,1)
      PRIORITIES:    MEDIUM              LOW

   CLASS LIST:       DI1W
      SERVICE TIMES: REX*DISCRETE(1,1)
      PRIORITIES:    LOW

QUEUE:L1 /* LBUS1 */
   TYPE: PS
   CLASS LIST:    FML1                FD1L1
   SERVICE TIMES: BEXM*DISCRETE(1,1) BEXD1*DISCRETE(1,1)
```

```
QUEUE:K1 /* CONTROLLER 1 */
   TYPE   CLASS LIST:    KI1                     KX1
   SERVICE TIMES: KEX*DISCRETE(1,1)     KEX*DISCRETE(1,1)

QUEUE:G /* GBUS */
   TYPE: PS
   CLASS LIST:    FMG                 FD1G
   SERVICE TIMES: BEXM*DISCRETE(1,1) BEXD1*DISCRETE(1,1)

   CLASS LIST:    FD2G
   SERVICE TIMES: BEXD2*DISCRETE(1,1)

QUEUE:K2 /* CONTROLLER 2 */
   TYPE   CLASS LIST:    KI2                  KX2
   SERVICE TIMES: KEX*DISCRETE(1,1)  KEX*DISCRETE(1,1)

QUEUE:L2 /* LBUS2 */
   TYPE: PS
   CLASS LIST:    FML2                FD1L2
   SERVICE TIMES: BEXM*DISCRETE(1,1) BEXD1*DISCRETE(1,1)

   CLASS LIST:    FD2L2
   SERVICE TIMES: BEXD2*DISCRETE(1,1)

QUEUE:M2 /* MEMORY REQUEST PROCESSOR 2 */
   TYPE   CLASS LIST:    MI2                 MX2
   SERVICE TIMES: REX*DISCRETE(1,1) REX*DISCRETE(1,1)

QUEUE:K3 /* CONTROLLER 3 */
   TYPE   CLASS LIST:    KI3                 KX3
   SERVICE TIMES: KEX*DISCRETE(1,1) KEX*DISCRETE(1,1)

QUEUE:L3 /* LBUS3 */
   TYPE: PS
   CLASS LIST:    FML3                FD2L3
   SERVICE TIMES: BEXM*DISCRETE(1,1) BEXD2*DISCRETE(1,1)

QUEUE:M3 /* MEMORY REQUEST PROCESSOR 3 */
   TYPE   CLASS LIST:    MI3                 MX3
   SERVICE TIMES: REX*DISCRETE(1,1) REX*DISCRETE(1,1)

QUEUE:D21 /* LEVEL 2 DEVICE 1 */
   TYPE   CLASS LIST:    DI21                 DX21
   SERVICE TIMES: DEX2*DISCRETE(1,1) DEX2*DISCRETE(1,1)

QUEUE:D22 /* LEVEL 2 DEVIE 2 */
   TYPE   CLASS LIST:    DI22                 DX22
   SERVICE TIMES: DEX2*DISCRETE(1,1) DEX2*DISCRETE(1,1)

QUEUE:D31 /* LEVEL 3 DEVICE 1 */
   TYPE   CLASS LIST:    DI31                 DX31
   SERVICE TIMES: DEX3*DISCRETE(1,1) DEX3*DISCRETE(1,1)
```

```
QUEUE:D32 /* LEVEL 3 DEVICE 2 */
   TYPE   CLASS LIST:    DI32              DX32
   SERVICE TIMES: DEX3*DISCRETE(1,1) DEX3*DISCRETE(1,1)

/************************************************/
/*   SET NODES FOR COLLECTING STATISTICS        */
/************************************************/


SET NODES:         SSTAT     /* SUMMARIZE STATISTICS */
ASSIGNMENT LIST:  SUMW = SUMW + CLOCK - JV(0)             ++
                  NTXN = NTXN + 1                         ++
                  MRESP = SUMW/NTXN

SET NODES:         STIME     /* SET START TIME */
ASSIGNMENT LIST:  JV(0) = CLOCK /* CURRENT SIMULATION TIME */


/******************************/
/*   FLOW UNBALANCED POINTS  */
/******************************/

SPLIT NODES: OVL11 SPACK2 SPACK3 SPOVH2 SPSTB1 SPSTOR1

/************************************************/
/*   DUMMY NODES TO CLARIFY ROUTING DEFINITIONS  */
/************************************************/

DUMMY NODES: ACK2 ACK21 ACK22 ACK3
DUMMY NODES: COMR COMW
DUMMY NODES: INL2 INL3
DUMMY NODES: NIN2 NOV11 NOV2
DUMMY NODES: OVF11 OVH2 OVL1 OVL2
DUMMY NODES:RRR21 RRR22 RRR31 RRR32 RTF2 RTF3 RTOK
DUMMY NODES: STB1 STB23 STOR1 STOR2 SWS21 SWS22 SWS31 SWS32
DUMMY NODES: SSS2 SSS21 SSS22 WWW1 WWW11

/************************************************/
/*              ROUTING DEFINITIONS             */
/************************************************/

CHAIN:TADP1L3

   TYPE:OPEN

   SOURCE LIST:S

   ARRIVAL TIMES:INTARRTIME

   :S -> SINK

   /*********************************************/
   /*   START FOR CPU TXNS                      */
   /*********************************************/
```

```
:STAR1 ->STIME -> WWW1 PRDI1R ; 1-PREAD  PREAD

:PRDI1R -> DI1R FML1(1) ; PIN1 1-PIN1

:DI1R -> SSTAT

:SSTAT -> STAR1  /* ACCUMULATE STATISTICS */

:FML1(1) -> COMR

/**********************************************/
/*   WRITE TRANSACTION                        */
/**********************************************/

:WWW1 -> PRDI1W -> SPSTB1

:SPSTB1 -> SSTAT STB1; SPLIT

:STB1 -> FD1L1(1) -> COMW

/**********************************************/
/*   COMMON CODE FOR READ TO LOWER LEVELS     */
/**********************************************/

:COMR -> KI1(1) -> FMG(1) -> KI2(1) -> FML2(1)

:FML2(1) -> MI2(1) -> INL2 NIN2  ; PIN2 1-PIN2

/**********************************************/
/*   DATA IS NOT FOUND IN LEVEL 2             */
/**********************************************/

:NIN2 -> FML2(2) -> KI2(2) -> FMG(2)

:FMG(2) -> KI3(1) -> FML3(1) -> MI3(1) -> INL3

/**********************************************/
/*   DATA IS FOUND IN LEVEL 2                 */
/**********************************************/

:INL2 -> FML2(3) -> RRR21 RRR22; .5 .5

/**********************/
/*   DATA IS IN D21   */
/**********************/

:RRR21 -> DI21

:DI21 -> FD1L2(1) -> RTF2

/**********************/
/*   DATA IS IN D22   */
/**********************/
```

```
:RRR22 -> DI22

:DI22 -> FD1L2(2) -> RTF2

/*********************************/
/*   READ THROUGH FROM LEVEL 2   */
/*********************************/

:RTF2 -> KX2(1)

:KX2(1) -> FD1G(1) -> STOR1

/*******************************************************/
/*   STORE DATA IN LEVEL 1 AS A RESULT OF READ THROUGH */
/*******************************************************/

:STOR1 -> KX1(1) -> WWW11

:WWW11 -> FD1L1(2) -> DX1

:DX1 -> NOV11 OVL11 ; 1-POV1 POV1

:NOV11 -> SSTAT

/*************************************************/
/*   OVERFLOW FROM LEVEL 1; END READ TXN;        */
/*   AT THE SAME TIME HANDLE THE OVERFLOW.       */
/*************************************************/

:OVL11 -> SSTAT OVF11; SPLIT

:OVF11 -> FML1(2) -> OVL1 -> KI1(2) -> FMG(3) -> KI2(3)

:KI2(3) -> FML2(4) -> MI2(2)-> SINK

/*************************************************/
/*   DATA IS FOUND IN LEVEL 3                    */
/*************************************************/

:INL3 -> FML3(2) -> RRR31 RRR32; .5 .5

:RRR31 -> DI31

:DI31 -> FD2L3(1) -> RTF3

:RRR32 -> DI32

:DI32 -> FD2L3(2) -> RTF3

/*********************************/
/*   READ THROUGH FROM LEVEL 3   */
/*********************************/
```

```
:RTF3 -> KX3(1) -> RTOK

:RTOK -> FD2G(1) -> SPSTOR1

:SPSTOR1 -> STOR1 STOR2; SPLIT

/**************************************************/
/*   READ-THROUGH TO LEVEL 2                      */
/**************************************************/

:STOR2 -> KX2(2)

:KX2(2) -> FD2L2(1)

:FD2L2(1) -> MX2(1) -> SPOVH2

:SPOVH2 -> SSS2 OVH2; SPLIT

:SSS2 -> SSS21 SSS22; .5 .5

/**********************/
/*   STORE INTO D21   */
/**********************/

:SSS21 -> FD2L2(2)

:FD2L2(2) -> DX21(1) -> SINK

/**********************/
/*   STORE INTO D22   */
/**********************/

:SSS22 -> FD2L2(3)

:FD2L2(3) -> DX22(1) -> SINK

:OVH2 -> NOV2 OVL2; 1-POV2 POV2

:NOV2 -> SINK

/**************************************************/
/*   HANDLE ANY OVERFLOW FROM LEVEL 2             */
/**************************************************/

:OVL2 -> FML2(5) -> KI2(4)-> FMG(4)

:FMG(4) -> KI3(2) -> FML3(3) -> MI3(2)-> SINK

/**************************************************/
/*   COMMON CODE FOR WRITE TO LOWER LEVELS        */
/**************************************************/

:COMW -> KX1(2)
```

```
:KX1(2) -> FD1G(2)

:FD1G(2) -> KX2(3)

:KX2(3) -> FD1L2(3)

:FD1L2(3) -> MX2(2) -> SWS21 SWS22; .5 .5

/***********************/
/*   SERVICED BY D21   */
/***********************/

:SWS21 -> FD1L2(4) -> DX21(2) -> FD2L2(4) -> SPACK2

:SPACK2 -> ACK2 STB23; SPLIT

:ACK2 -> FML2(6) -> ACK21

/***********************/
/*   SERVICED BY D22   */
/***********************/

:SWS22 -> FD1L2(5) -> DX22(2) -> FD2L2(5) -> SPACK3

:SPACK3 -> ACK3 STB23; SPLIT

:ACK3 -> FML2(7) -> ACK21

/*********************************************/
/*   STORE-BEHIND FROM LEVEL 2 TO LEVEL 3   */
/*********************************************/

:STB23 -> KX2(4) -> FD2G(2) -> KX3(2) -> FD2L3(3) -> MX3

:MX3 -> SWS31 SWS32; .5 .5

/***********************/
/*   SERVICED BY D31   */
/***********************/

:SWS31 -> FD2L3(4) -> DX31

:DX31 -> FML3(4)

/***********************/
/*   SERVICED BY D32   */
/***********************/

:SWS32 -> FD2L3(5) -> DX32

:DX32 -> FML3(4)

:FML3(4) -> ACK22
```

```
/**********************************************/
/*   ACKNOWLEDGEMENT FROM LEVEL 3 TO LEVEL 2   */
/**********************************************/

:ACK22 -> KI3(3) -> FMG(5) -> KI2(5)

:KI2(5) -> FML2(8) -> MI2(3)

:MI2(3) -> FML2(9) -> ACK21

/**********************************************/
/*   ACKNOWLEDGEMENT FROM LEVEL 2 TO LEVEL 1   */
/**********************************************/

:ACK21 -> KI2(6)-> FMG(6) -> KI1(3)

:KI1(3) -> FML1(3) -> DI1W -> SINK

CONFIDENCE INTERVAL METHOD:NONE

INITIAL STATE DEFINITION -

CHAIN:TADP1L3

NODE LIST:    STAR1

   INIT POP:  MAXMP

RUN LIMITS -

   SIMULATED TIME: SIM_TIME

LIMIT - CP SECONDS: CPU_SEC

TRACE:NO

END
```

Appendix V:

Listing of Simulation Results of P1L3 Model using RESQ

```
MODEL:TADP1L3
CPU_SEC:100
HIGH:1          /* HIGH PRIORITY */
LOW:1           /* LOW PRIORITY */
MAXMP:20        /* MAX DEGREE OF MULTIPROGRAMMING */
MEDIUM:1        /* MEDIUM PRIORITY */
PIN1:.7         /* PROBABILITY IN LEVEL 1 */
PIN2:.7         /* PROBABILITY IN LEVEL 2 */
PIN3:1.0        /* PROBABILITY IN LEVEL 3 */
POV1:.5         /* PROBABILITY OF OVERFLOW LEVEL 1 */
POV2:.5         /* PROBABILITY OF OVERFLOW LEVEL 2 */
PREAD:.7        /* PROPORTION OF READ REQUESTS */
SIM_TIME:100000000
RUN END: CPU LIMIT
NO ERRORS DETECTED DURING SIMULATION.

                  SIMULATED TIME:     3.2434E+06
                        CPU TIME:         100.34
                NUMBER OF EVENTS:          97358

WHAT:GV

ELEMENT         FINAL VALUES OF GLOBAL VARIABLES
CLOCK           3.2434E+06
MRESP           1.0922E+04
NTXN            5878.00000
SUMW            6.4197E+07

ELEMENT         UTILIZATION
START           0.00000
D1              0.64014
L1              0.25110
K1              0.25110
G               0.80593
K2              0.36909
L2              0.97544
M2              0.34360
K3              0.12900
L3              0.98012
M3              0.13418
D21             0.45478
D22             0.45354
D31             0.62713
D32             0.60202

CONTINUE RUN:YES

LIMIT - CP SECONDS:200

RUN END: CPU LIMIT
RUN END: CPU LIMIT
NO ERRORS DETECTED DURING SIMULATION.
```

```
                 SIMULATED TIME:     6.5962E+06
                       CPU TIME:         200.11
               NUMBER OF EVENTS:         193724

WHAT:GV

ELEMENT         FINAL VALUES OF GLOBAL VARIABLES
CLOCK           6.5962E+06
MRESP           1.1560E+04
NTXN            1.1331E+04
SUMW            1.3099E+08

ELEMENT         UTILIZATION
START           0.00000
D1              0.61495
 PRDI1R          0.23862
 PRDI1W          0.05274
 DI1R            0.08316
 DX1             0.03588
 DI1W            0.20454
L1              0.24489
 FD1L1(1)        0.05139
 FD1L1(2)        0.03611
 FML1(1)         0.03323
 FML1(2)         0.01680
 FML1(3)         0.10735
K1              0.24489
 KI1(1)          0.03267
 KI1(2)          0.01693
 KI1(3)          0.10652
 KX1(1)          0.03624
 KX1(2)          0.05253
G               0.80004
 FD1G(1)         0.02092
 FD1G(2)         0.04533
 FD2G(1)         0.06923
 FD2G(2)         0.44407
 FMG(1)          0.02961
 FMG(2)          9.2314E-03
 FMG(3)          0.01544
 FMG(4)          4.5884E-03
 FMG(5)          0.04740
 FMG(6)          0.11421
K2              0.36336
 KI2(1)          0.03347
 KI2(2)          0.01051
 KI2(3)          0.01648
 KI2(4)          5.0560E-03
 KI2(5)          0.05120
 KI2(6)          0.10404
 KX2(1)          0.02485
 KX2(2)          0.01040
 KX2(3)          0.05351
```

KX2(4)	0.05385
L2	0.96368
FD1L2(1)	0.01228
FD1L2(2)	0.01321
FD1L2(3)	0.05359
FD1L2(4)	0.02740
FD1L2(5)	0.02654
FD2L2(1)	0.07878
FD2L2(2)	0.03998
FD2L2(3)	0.03808
FD2L2(4)	0.21739
FD2L2(5)	0.21189
FML2(1)	0.03591
FML2(2)	0.01042
FML2(3)	0.02546
FML2(4)	0.01785
FML2(5)	4.9972E-03
FML2(6)	0.02657
FML2(7)	0.02590
FML2(8)	0.04877
FML2(9)	0.04865
M2	0.33513
MI2(1)	0.07100
MI2(2)	0.03310
MI2(3)	0.09836
MX2(1)	0.01966
MX2(2)	0.11301
K3	0.12901
KI3(1)	0.01046
KI3(2)	5.0480E-03
KI3(3)	0.05056
KX3(1)	0.01016
KX3(2)	0.05278
L3	0.98519
FD2L3(1)	0.04204
FD2L3(2)	0.03972
FD2L3(3)	0.41405
FD2L3(4)	0.21038
FD2L3(5)	0.20271
FML3(1)	0.01036
FML3(2)	0.01035
FML3(3)	5.0903E-03
FML3(4)	0.05049
M3	0.13459
MI3(1)	0.02032
MI3(2)	9.8308E-03
MX3	0.10445
D21	0.44329
DX21(1)	0.04362
DX21(2)	0.28420
DI21	0.11547
D22	0.43920
DX22(1)	0.04255
DX22(2)	0.27342

DI22	0.12323
D31	0.62421
DI31	0.09737
DX31	0.52684
D32	0.60672
DI32	0.08667
DX32	0.52005

ELEMENT	THROUGHPUT
START	1.7209E-03
D1	3.9337E-03
PRDI1R	1.1931E-03
PRDI1W	5.2743E-04
DI1R	8.3154E-04
DX1	3.5885E-04
DI1W	1.0227E-03
L1	2.4489E-03
FD1L1(1)	5.2743E-04
FD1L1(2)	3.5885E-04
FML1(1)	3.6142E-04
FML1(2)	1.7829E-04
FML1(3)	1.0229E-03
K1	2.4489E-03
KI1(1)	3.6142E-04
KI1(2)	1.7829E-04
KI1(3)	1.0229E-03
KX1(1)	3.5885E-04
KX1(2)	5.2743E-04
G	3.6335E-03
FD1G(1)	2.5333E-04
FD1G(2)	5.2743E-04
FD2G(1)	1.0552E-04
FD2G(2)	5.1833E-04
FMG(1)	3.6142E-04
FMG(2)	1.0718E-04
FMG(3)	1.7829E-04
FMG(4)	5.1545E-05
FMG(5)	5.0757E-04
FMG(6)	1.0229E-03
K2	3.6335E-03
KI2(1)	3.6142E-04
KI2(2)	1.0718E-04
KI2(3)	1.7829E-04
KI2(4)	5.1545E-05
KI2(5)	5.0757E-04
KI2(6)	1.0229E-03
KX2(1)	2.5333E-04
KX2(2)	1.0552E-04
KX2(3)	5.2743E-04
KX2(4)	5.1833E-04
L2	4.5116E-03
FD1L2(1)	1.2386E-04
FD1L2(2)	1.2947E-04

FD1L2(3)	5.2606E-04
FD1L2(4)	2.6697E-04
FD1L2(5)	2.5818E-04
FD2L2(1)	1.0430E-04
FD2L2(2)	5.2455E-05
FD2L2(3)	5.1545E-05
FD2L2(4)	2.6364E-04
FD2L2(5)	2.5485E-04
FML2(1)	3.6097E-04
FML2(2)	1.0718E-04
FML2(3)	2.5333E-04
FML2(4)	1.7813E-04
FML2(5)	5.1545E-05
FML2(6)	2.6349E-04
FML2(7)	2.5439E-04
FML2(8)	5.0620E-04
FML2(9)	5.0499E-04
M2	1.6757E-03
MI2(1)	3.6097E-04
MI2(2)	1.7813E-04
MI2(3)	5.0620E-04
MX2(1)	1.0430E-04
MX2(2)	5.2606E-04
K3	1.2901E-03
KI3(1)	1.0718E-04
KI3(2)	5.1545E-05
KI3(3)	5.0757E-04
KX3(1)	1.0552E-04
KX3(2)	5.1833E-04
L3	1.9019E-03
FD2L3(1)	5.3213E-05
FD2L3(2)	5.2303E-05
FD2L3(3)	5.1454E-04
FD2L3(4)	2.5803E-04
FD2L3(5)	2.5060E-04
FML3(1)	1.0703E-04
FML3(2)	1.0703E-04
FML3(3)	5.1545E-05
FML3(4)	5.0757E-04
M3	6.7297E-04
MI3(1)	1.0703E-04
MI3(2)	5.1545E-05
MX3	5.1439E-04
D21	4.4329E-04
DX21(1)	5.2455E-05
DX21(2)	2.6697E-04
DI21	1.2386E-04
D22	4.3920E-04
DX22(1)	5.1545E-05
DX22(2)	2.5818E-04
DI22	1.2947E-04
D31	3.1200E-04
DI31	5.4122E-05
DX31	2.5788E-04

D32	3.0336E-04
DI32	5.2758E-05
DX32	2.5060E-04
SSTAT	1.7178E-03
STIME	1.7209E-03
OVL11	1.7829E-04
SPACK2	2.6364E-04
SPACK3	2.5485E-04
SPOVH2	1.0430E-04
SPSTB1	5.2743E-04
SPSTOR1	1.0552E-04
ACK2	2.6364E-04
ACK21	1.0229E-03
ACK22	5.0757E-04
ACK3	2.5485E-04
COMR	3.6142E-04
COMW	5.2743E-04
INL2	2.5363E-04
INL3	1.0703E-04
NIN2	1.0734E-04
NOV11	1.8056E-04
NOV2	5.2758E-05
OVF11	1.7829E-04
OVH2	1.0430E-04
OVL1	1.7829E-04
OVL2	5.1545E-05
RRR21	1.2386E-04
RRR22	1.2947E-04
RRR31	5.4274E-05
RRR32	5.2758E-05
RTF2	2.5333E-04
RTF3	1.0552E-04
RTOK	1.0552E-04
STB1	5.2743E-04
STB23	5.1848E-04
STOR1	3.5885E-04
STOR2	1.0552E-04
SWS21	2.6728E-04
SWS22	2.5879E-04
SWS31	2.6136E-04
SWS32	2.5303E-04
SSS2	1.0430E-04
SSS21	5.2758E-05
SSS22	5.1545E-05
WWW1	5.2743E-04
WWW11	3.5885E-04
SINK	1.4092E-03

ELEMENT	MEAN QUEUE LENGTH
START	0.00000
D1	1.37406
PRDI1R	0.47966
PRDI1W	0.15958

DI1R	0.26735
DX1	0.09235
DI1W	0.37512
L1	0.31366
FD1L1(1)	0.06583
FD1L1(2)	0.04625
FML1(1)	0.04257
FML1(2)	0.02151
FML1(3)	0.13751
K1	0.32097
KI1(1)	0.04282
KI1(2)	0.02218
KI1(3)	0.13962
KX1(1)	0.04749
KX1(2)	0.06885
G	7.91954
FD1G(1)	0.20707
FD1G(2)	0.44875
FD2G(1)	0.68531
FD2G(2)	4.39582
FMG(1)	0.29313
FMG(2)	0.09138
FMG(3)	0.15287
FMG(4)	0.04542
FMG(5)	0.46925
FMG(6)	1.13054
K2	0.59288
KI2(1)	0.05462
KI2(2)	0.01715
KI2(3)	0.02689
KI2(4)	8.2497E-03
KI2(5)	0.08353
KI2(6)	0.16976
KX2(1)	0.04054
KX2(2)	0.01696
KX2(3)	0.08731
KX2(4)	0.08787
L2	76.70413
FD1L2(1)	0.97781
FD1L2(2)	1.05140
FD1L2(3)	4.26544
FD1L2(4)	2.18092
FD1L2(5)	2.11235
FD2L2(1)	6.27031
FD2L2(2)	3.18256
FD2L2(3)	3.03127
FD2L2(4)	17.30345
FD2L2(5)	16.86565
FML2(1)	2.85818
FML2(2)	0.82971
FML2(3)	2.02629
FML2(4)	1.42084
FML2(5)	0.39775
FML2(6)	2.11499

FML2(7)	2.06155
FML2(8)	3.88159
FML2(9)	3.87210
M2	0.55846
MI2(1)	0.11832
MI2(2)	0.05516
MI2(3)	0.16390
MX2(1)	0.03276
MX2(2)	0.18832
K3	0.15156
KI3(1)	0.01229
KI3(2)	5.9300E-03
KI3(3)	0.05939
KX3(1)	0.01194
KX3(2)	0.06200
L3	79.83951
FD2L3(1)	3.40692
FD2L3(2)	3.21859
FD2L3(3)	33.55435
FD2L3(4)	17.04948
FD2L3(5)	16.42714
FML3(1)	0.83991
FML3(2)	0.83911
FML3(3)	0.41251
FML3(4)	4.09150
M3	0.16638
MI3(1)	0.02511
MI3(2)	0.01215
MX3	0.12911
D21	1.08901
DX21(1)	0.10716
DX21(2)	0.69817
DI21	0.28367
D22	1.03493
DX22(1)	0.10026
DX22(2)	0.64430
DI22	0.29038
D31	2.40771
DI31	0.37557
DX31	2.03213
D32	1.87286
DI32	0.26754
DX32	1.60531

ELEMENT	MEAN QUEUEING TIME
START	0.00000
D1	349.29346
PRDI1R	402.00757
PRDI1W	302.55811
DI1R	321.47656
DX1	257.36450
DI1W	366.77100
L1	128.08629

FD1L1(1)	124.80728
FD1L1(2)	128.88383
FML1(1)	117.77933
FML1(2)	120.67012
FML1(3)	134.43176
K1	131.06851
KI1(1)	118.46901
KI1(2)	124.43440
KI1(3)	136.49586
KX1(1)	132.34869
KX1(2)	130.54839
G	2179.60156
FD1G(1)	817.38452
FD1G(2)	850.83374
FD2G(1)	6494.85156
FD2G(2)	8480.69922
FMG(1)	811.04736
FMG(2)	852.56104
FMG(3)	857.46899
FMG(4)	881.17114
FMG(5)	924.49634
FMG(6)	1105.26001
K2	163.16872
KI2(1)	151.11467
KI2(2)	159.99593
KI2(3)	150.82721
KI2(4)	160.04846
KI2(5)	164.57693
KI2(6)	165.96339
KX2(1)	160.03802
KX2(2)	160.76300
KX2(3)	165.53809
KX2(4)	169.50005
L2	1.6934E+04
FD1L2(1)	7894.45703
FD1L2(2)	8120.82422
FD1L2(3)	8098.81250
FD1L2(4)	8166.73438
FD1L2(5)	8177.44922
FD2L2(1)	5.9777E+04
FD2L2(2)	6.0510E+04
FD2L2(3)	5.8808E+04
FD2L2(4)	6.5208E+04
FD2L2(5)	6.5698E+04
FML2(1)	7913.83984
FML2(2)	7734.85547
FML2(3)	7990.48828
FML2(4)	7975.50781
FML2(5)	7716.59766
FML2(6)	8024.76172
FML2(7)	8097.78125
FML2(8)	7655.10938
FML2(9)	7655.71875
M2	333.27661

MI2(1)	327.77686
MI2(2)	309.66992
MI2(3)	323.78540
MX2(1)	314.11816
MX2(2)	357.97559
K3	117.47263
KI3(1)	114.68506
KI3(2)	115.04561
KI3(3)	117.01743
KX3(1)	113.13005
KX3(2)	119.62016
L3	4.1748E+04
FD2L3(1)	6.3345E+04
FD2L3(2)	6.1056E+04
FD2L3(3)	6.4902E+04
FD2L3(4)	6.5629E+04
FD2L3(5)	6.5153E+04
FML3(1)	7836.05078
FML3(2)	7839.79688
FML3(3)	8002.93359
FML3(4)	8055.91797
M3	247.23100
MI3(1)	234.63078
MI3(2)	235.76187
MX3	251.00208
D21	2456.66089
DX21(1)	2042.97485
DX21(2)	2615.14063
DI21	2290.26245
D22	2356.42749
DX22(1)	1945.03540
DX22(2)	2495.52588
DI22	2242.83081
D31	7715.78906
DI31	6939.23828
DX31	7878.76953
D32	6173.74609
DI32	5071.14063
DX32	6405.87109

ELEMENT	MAXIMUM QUEUE LENGTH
START	20
D1	20
PRDI1R	14
PRDI1W	6
DI1R	10
DX1	3
DI1W	8
L1	8
FD1L1(1)	5
FD1L1(2)	3
FML1(1)	2
FML1(2)	2

FML1(3)	6
K1	8
KI1(1)	2
KI1(2)	2
KI1(3)	6
KX1(1)	4
KX1(2)	5
G	72
FD1G(1)	5
FD1G(2)	9
FD2G(1)	7
FD2G(2)	40
FMG(1)	5
FMG(2)	3
FMG(3)	5
FMG(4)	2
FMG(5)	8
FMG(6)	23
K2	11
KI2(1)	3
KI2(2)	2
KI2(3)	2
KI2(4)	2
KI2(5)	4
KI2(6)	5
KX2(1)	3
KX2(2)	2
KX2(3)	5
KX2(4)	5
L2	197
FD1L2(1)	7
FD1L2(2)	8
FD1L2(3)	22
FD1L2(4)	15
FD1L2(5)	14
FD2L2(1)	19
FD2L2(2)	13
FD2L2(3)	11
FD2L2(4)	58
FD2L2(5)	49
FML2(1)	13
FML2(2)	10
FML2(3)	11
FML2(4)	7
FML2(5)	4
FML2(6)	15
FML2(7)	14
FML2(8)	18
FML2(9)	18
M2	10
MI2(1)	4
MI2(2)	3
MI2(3)	5
MX2(1)	2

MX2(2)	7
K3	5
KI3(1)	2
KI3(2)	2
KI3(3)	3
KX3(1)	2
KX3(2)	5
L3	189
FD2L3(1)	11
FD2L3(2)	10
FD2L3(3)	87
FD2L3(4)	55
FD2L3(5)	48
FML3(1)	6
FML3(2)	6
FML3(3)	4
FML3(4)	25
M3	6
MI3(1)	3
MI3(2)	2
MX3	6
D21	13
DX21(1)	3
DX21(2)	11
DI21	5
D22	16
DX22(1)	3
DX22(2)	11
DI22	5
D31	25
DI31	4
DX31	23
D32	19
DI32	5
DX32	17

ELEMENT	MAXIMUM QUEUEING TIME
START	0.00000
D1	3400.00000
PRDI1R	3400.00000
PRDI1W	3400.00000
DI1R	3200.00000
DX1	2808.18140
DI1W	2650.81152
L1	581.16846
FD1L1(1)	581.16846
FD1L1(2)	549.06543
FML1(1)	513.50342
FML1(2)	452.17725
FML1(3)	565.42651
K1	665.65845
KI1(1)	441.81323
KI1(2)	645.96436

KI1(3)	642.84961
KX1(1)	665.65845
KX1(2)	661.41357
G	4.0380E+04
FD1G(1)	6300.22266
FD1G(2)	6423.61719
FD2G(1)	3.9938E+04
FD2G(2)	4.0380E+04
FMG(1)	6252.17578
FMG(2)	5493.85938
FMG(3)	6364.11328
FMG(4)	5133.42188
FMG(5)	6438.33984
FMG(6)	6431.50781
K2	910.31616
KI2(1)	880.74829
KI2(2)	674.80884
KI2(3)	626.01733
KI2(4)	802.71582
KI2(5)	709.76807
KI2(6)	904.23169
KX2(1)	715.29907
KX2(2)	719.34521
KX2(3)	909.34375
KX2(4)	910.31616
L2	1.4167E+05
FD1L2(1)	1.8979E+04
FD1L2(2)	1.9164E+04
FD1L2(3)	1.9123E+04
FD1L2(4)	1.9151E+04
FD1L2(5)	1.9158E+04
FD2L2(1)	1.4165E+05
FD2L2(2)	1.4167E+05
FD2L2(3)	1.3825E+05
FD2L2(4)	1.4166E+05
FD2L2(5)	1.4166E+05
FML2(1)	1.9158E+04
FML2(2)	1.8904E+04
FML2(3)	1.9106E+04
FML2(4)	1.9132E+04
FML2(5)	1.8413E+04
FML2(6)	1.9166E+04
FML2(7)	1.9020E+04
FML2(8)	1.9167E+04
FML2(9)	1.9167E+04
M2	1634.19556
MI2(1)	1614.36963
MI2(2)	1634.19556
MI2(3)	1577.35400
MX2(1)	1334.56201
MX2(2)	1612.03955
K3	401.86743
KI3(1)	374.20288
KI3(2)	331.81982

KI3(3)	401.86743
KX3(1)	307.38501
KX3(2)	374.37158
L3	1.4121E+05
FD2L3(1)	1.4038E+05
FD2L3(2)	1.4086E+05
FD2L3(3)	1.4121E+05
FD2L3(4)	1.4120E+05
FD2L3(5)	1.4121E+05
FML3(1)	1.8381E+04
FML3(2)	1.8427E+04
FML3(3)	1.8039E+04
FML3(4)	1.8529E+04
M3	950.95117
MI3(1)	944.80371
MI3(2)	698.56274
MX3	950.95117
D21	1.0973E+04
DX21(1)	9270.82031
DX21(2)	1.0973E+04
DI21	1.0973E+04
D22	1.3226E+04
DX22(1)	9103.28906
DX22(2)	1.3222E+04
DI22	1.3226E+04
D31	4.0320E+04
DI31	4.0151E+04
DX31	4.0320E+04
D32	3.0827E+04
DI32	3.0772E+04
DX32	3.0827E+04

ELEMENT	NUMBER OF DEPARTURES
START	11351
D1	25947
PRDI1R	7870
PRDI1W	3479
DI1R	5485
DX1	2367
DI1W	6746
L1	16153
FD1L1(1)	3479
FD1L1(2)	2367
FML1(1)	2384
FML1(2)	1176
FML1(3)	6747
K1	16153
KI1(1)	2384
KI1(2)	1176
KI1(3)	6747
KX1(1)	2367
KX1(2)	3479
G	23967

FD1G(1)	1671
FD1G(2)	3479
FD2G(1)	696
FD2G(2)	3419
FMG(1)	2384
FMG(2)	707
FMG(3)	1176
FMG(4)	340
FMG(5)	3348
FMG(6)	6747
K2	23967
KI2(1)	2384
KI2(2)	707
KI2(3)	1176
KI2(4)	340
KI2(5)	3348
KI2(6)	6747
KX2(1)	1671
KX2(2)	696
KX2(3)	3479
KX2(4)	3419
L2	29759
FD1L2(1)	817
FD1L2(2)	854
FD1L2(3)	3470
FD1L2(4)	1761
FD1L2(5)	1703
FD2L2(1)	688
FD2L2(2)	346
FD2L2(3)	340
FD2L2(4)	1739
FD2L2(5)	1681
FML2(1)	2381
FML2(2)	707
FML2(3)	1671
FML2(4)	1175
FML2(5)	340
FML2(6)	1738
FML2(7)	1678
FML2(8)	3339
FML2(9)	3331
M2	11053
MI2(1)	2381
MI2(2)	1175
MI2(3)	3339
MX2(1)	688
MX2(2)	3470
K3	8510
KI3(1)	707
KI3(2)	340
KI3(3)	3348
KX3(1)	696
KX3(2)	3419
L3	12545

FD2L3(1)	351
FD2L3(2)	345
FD2L3(3)	3394
FD2L3(4)	1702
FD2L3(5)	1653
FML3(1)	706
FML3(2)	706
FML3(3)	340
FML3(4)	3348
M3	4439
MI3(1)	706
MI3(2)	340
MX3	3393
D21	2924
DX21(1)	346
DX21(2)	1761
DI21	817
D22	2897
DX22(1)	340
DX22(2)	1703
DI22	854
D31	2058
DI31	357
DX31	1701
D32	2001
DI32	348
DX32	1653
SSTAT	11331
STIME	11351
OVL11	1176
SPACK2	1739
SPACK3	1681
SPOVH2	688
SPSTB1	3479
SPSTOR1	696
ACK2	1739
ACK21	6747
ACK22	3348
ACK3	1681
COMR	2384
COMW	3479
INL2	1673
INL3	706
NIN2	708
NOV11	1191
NOV2	348
OVF11	1176
OVH2	688
OVL1	1176
OVL2	340
RRR21	817
RRR22	854
RRR31	358
RRR32	348

RTF2	1671
RTF3	696
RTOK	696
STB1	3479
STB23	3420
STOR1	2367
STOR2	696
SWS21	1763
SWS22	1707
SWS31	1724
SWS32	1669
SSS2	688
SSS21	348
SSS22	340
WWW1	3479
WWW11	2367
SINK	9295

ELEMENT	FINAL LENGTHS
START	0
D1	4
PRDI1R	2
PRDI1W	0
DI1R	1
DX1	0
DI1W	1
L1	0
FD1L1(1)	0
FD1L1(2)	0
FML1(1)	0
FML1(2)	0
FML1(3)	0
K1	0
KI1(1)	0
KI1(2)	0
KI1(3)	0
KX1(1)	0
KX1(2)	0
G	0
FD1G(1)	0
FD1G(2)	0
FD2G(1)	0
FD2G(2)	0
FMG(1)	0
FMG(2)	0
FMG(3)	0
FMG(4)	0
FMG(5)	0
FMG(6)	0
K2	1
KI2(1)	0
KI2(2)	0
KI2(3)	0

KI2(4)	0
KI2(5)	0
KI2(6)	0
KX2(1)	0
KX2(2)	0
KX2(3)	0
KX2(4)	1
L2	97
FD1L2(1)	0
FD1L2(2)	0
FD1L2(3)	9
FD1L2(4)	2
FD1L2(5)	4
FD2L2(1)	8
FD2L2(2)	2
FD2L2(3)	0
FD2L2(4)	22
FD2L2(5)	22
FML2(1)	3
FML2(2)	1
FML2(3)	2
FML2(4)	1
FML2(5)	0
FML2(6)	1
FML2(7)	3
FML2(8)	9
FML2(9)	8
M2	0
MI2(1)	0
MI2(2)	0
MI2(3)	0
MX2(1)	0
MX2(2)	0
K3	0
KI3(1)	0
KI3(2)	0
KI3(3)	0
KX3(1)	0
KX3(2)	0
L3	79
FD2L3(1)	6
FD2L3(2)	3
FD2L3(3)	25
FD2L3(4)	22
FD2L3(5)	16
FML3(1)	1
FML3(2)	0
FML3(3)	0
FML3(4)	6
M3	1
MI3(1)	0
MI3(2)	0
MX3	1
D21	0

DX21(1)	0
DX21(2)	0
DI21	0
D22	0
DX22(1)	0
DX22(2)	0
DI22	0
D31	2
DI31	1
DX31	1
D32	0
DI32	0
DX32	0

ELEMENT	FINAL VALUES OF GLOBAL VARIABLES
CLOCK	6.5962E+06
MRESP	1.1560E+04
NTXN	1.1331E+04
SUMW	1.3099E+08

ELEMENT	MEAN SERVICE TIMES
START	0.00000
D1	156.33020
PRDI1R	199.99998
PRDI1W	100.00000
DI1R	100.00000
DX1	100.00000
DI1W	200.00000
L1	99.99998
FD1L1(1)	97.44000
FD1L1(2)	100.62267
FML1(1)	91.95311
FML1(2)	94.21002
FML1(3)	104.95406
K1	99.99998
KI1(1)	90.38708
KI1(2)	94.93843
KI1(3)	104.14084
KX1(1)	100.97672
KX1(2)	99.60315
G	220.18608
FD1G(1)	82.57321
FD1G(2)	85.95229
FD2G(1)	656.11816
FD2G(2)	856.73071
FMG(1)	81.93301
FMG(2)	86.12679
FMG(3)	86.62259
FMG(4)	89.01701
FMG(5)	93.39378
FMG(6)	111.65474
K2	100.00218

KI2(1)	92.61331
KI2(2)	98.05634
KI2(3)	92.43712
KI2(4)	98.08853
KI2(5)	100.86389
KI2(6)	101.71361
KX2(1)	98.08214
KX2(2)	98.52646
KX2(3)	101.45294
KX2(4)	103.89050
L2	213.60301
FD1L2(1)	99.18326
FD1L2(2)	102.02724
FD1L2(3)	101.86885
FD1L2(4)	102.63304
FD1L2(5)	102.79163
FD2L2(1)	755.27856
FD2L2(2)	762.26611
FD2L2(3)	738.84253
FD2L2(4)	824.59375
FD2L2(5)	831.46167
FML2(1)	99.48019
FML2(2)	97.25497
FML2(3)	100.49214
FML2(4)	100.21051
FML2(5)	96.94868
FML2(6)	100.84735
FML2(7)	101.81456
FML2(8)	96.33859
FML2(9)	96.33372
M2	199.99998
MI2(1)	196.69955
MI2(2)	185.83353
MI2(3)	194.30434
MX2(1)	188.50291
MX2(2)	214.82181
K3	99.99998
KI3(1)	97.62704
KI3(2)	97.93398
KI3(3)	99.61250
KX3(1)	96.30333
KX3(2)	101.82811
L3	518.01343
FD2L3(1)	790.03857
FD2L3(2)	759.34595
FD2L3(3)	804.69409
FD2L3(4)	815.35303
FD2L3(5)	808.87769
FML3(1)	96.83249
FML3(2)	96.74039
FML3(3)	98.75346
FML3(4)	99.46976
M3	199.99998
MI3(1)	189.80692

MI3(2)	190.72194
MX3	203.05064
D21	999.99976
DX21(1)	831.60620
DX21(2)	1064.51001
DI21	932.26636
D22	999.99976
DX22(1)	825.41699
DX22(2)	1059.02930
DI22	951.79272
D31	2000.65771
DI31	1799.04370
DX31	2042.97192
D32	1999.99976
DI32	1642.80933
DX32	2075.19800

Appendix VI:

Listing of Analytic Results of P1L3 Model using TAD

This listing is generated by TAD for the P1L3 model presented in Chapter VI.3.2.. It also serves a s a sample session for those interested in using TAD. The *italic* font, as shown in the appendix, indicates the responses of the user while the regular font indicates the output from TAD.

```
****************************************
***                                  ***
***     INFOPLEX TAD VERSION 1.0     ***
***                                  ***
***  A TOOL FOR ARCHITECTURAL DESIGN ***
***                                  ***
***          NOVEMBER 1983           ***
***                                  ***
****************************************
```

IS THIS A NEW MODEL? CONFIRM YES/NO:
NO   S A NEW MODEL? CONFIRM YES/NO: NO

ENTER THE OLD MODEL'S NAME!
TBALANCED79.P13

NUMBER OF SERVICE FACILITIES IS: 14

LEVEL 1 LOCAL MEMORY SERVICE TIME IS: 100 ns.

BUS MESSAGE SERVICE TIME IS: 100 ns.

ADJUST PAPER IF NECESSARY; TYPE YES WHEN READY!
YES

```
               QQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQ
                                       Q
                                       Q
                                 ------------
                                 |          |
   1 GBUS                        |   1 PE   |
                                 |  200 ns  |
                                 |          |
                                 ------------
                                       |
  |            ------------------------------------------------
  |                 |                                            1 LBUS
  |                 |                                            100 ns
  |            ------------
  |            |          |
  |------------|   1 GC   |
  |            |  100 ns  |
  |            |          |
  |            ------------
  |                                LEVEL 1
  |
  |            ------------------------------------------------
  |                 |                  |                  |      1 LBUS
  |                 |                  |                  |      800 ns
  |            ------------       ------------       ------------
```

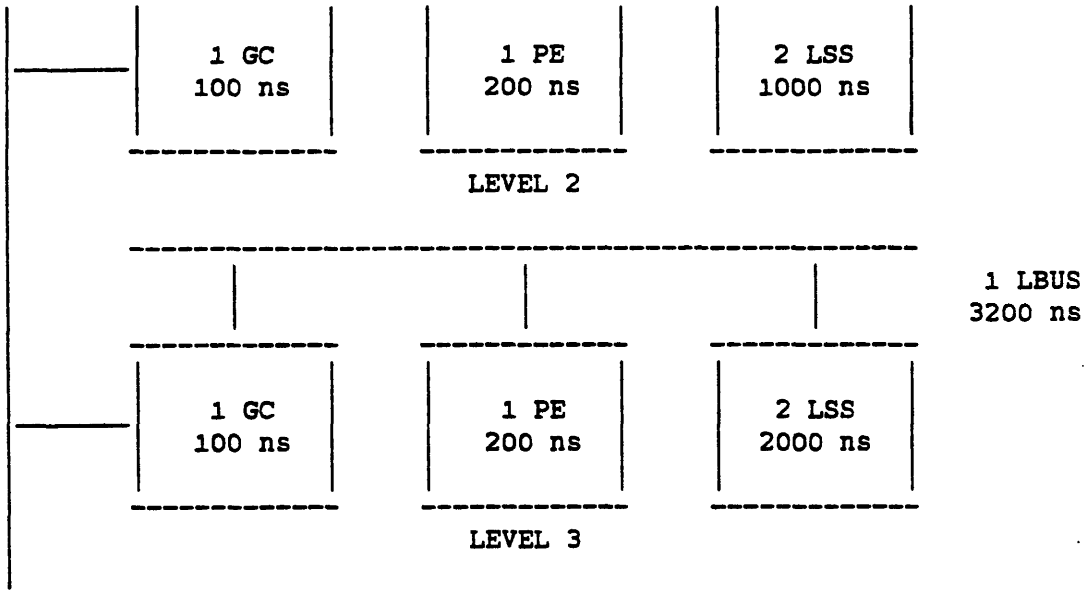


FIG-1: NUMBER OF SERVICE FACILITIES AND THEIR SERVICE TIMES.

THE PROBABILITY OF OVERFLOW LEVEL 1 IS: .5.
THE PROBABILITY OF OVERFLOW LEVEL 2 IS: .5.
THE PROBABILITY OF OVERFLOW LEVEL 3 IS: .5.

DO YOU WANT TO SAVE THE MODEL? CONFIRM YES/NO:
NO

DO YOU WANT TO AUDIT THE VISIT-RATIO REPORT? CONFIRM YES/NO
YES
****
YOU CAN SELECT THE COMBINATION OF POLICIES
BY ENTERING THE SUM OF THE POLICY NUMBERS BELOW:

10000 OPEN; 20000 CLOSED;

1000 PERCOLATE; 2000 PARALLEL;

100 RETRANSMIT; 200 RESERVE SPACE;

10 A (LOCALITY,READ%) POINT; 20 A LOCALITY SET GIVEN A READ%;
1 EQUAL PRIORITY; 2 STB LOW PRIORITY;

```
********************************************************************************
THE CURRENT COMBINATION OF POLICIES IS 11111 :
OPEN, PERCOLATE, RETRANSMIT, A (LOCALITY,READ%) POINT, AND EQUAL PRIORITY.
********************************************************************************


IS THIS WHAT YOU WANT? CONFIRM YES/NO:
NO


ENTER THE SUM OF THE COMBINATION OF POLICIES! 21111
********************************************************************************
YOU CAN SELECT THE COMBINATION OF POLICIES
BY ENTERING THE SUM OF THE POLICY NUMBERS BELOW:


10000 OPEN;           20000 CLOSED;

1000 PERCOLATE;       2000 PARALLEL;

100 RETRANSMIT;       200 RESERVE SPACE;

10 A (LOCALITY,READ%) POINT;             20 A LOCALITY SET GIVEN A READ%;
1 EQUAL PRIORITY;     2 STB LOW PRIORITY;


********************************************************************************
THE CURRENT COMBINATION OF POLICIES IS 21111 :
CLOSED, PERCOLATE, RETRANSMIT, A (LOCALITY,READ%) POINT, AND EQUAL PRIORITY.
********************************************************************************


IS THIS WHAT YOU WANT? CONFIRM YES/NO:
YES


ENTER A LOCALITY(ASSUME THE SAME ACROSS LEVELS)!
.7


ENTER READ%!
.7


ENTER THE POPULATION IN THE CLOSED CHAIN!
20


CHECK IN DSH LEVEL ONE PE.
--------------------------
```

NUMBER OF FACILITIES		LEVEL	VISIT-RATIO	SERVICE-TIME	VS-PRODUCT	CHAIN-TYPE
1	PE	1	.70000	200.000	140.0	1

READ-THROUGH-MESSAGE STOPS WHEN DATA IS FOUND;
IT'S FOLLOWED BY READ-THROUGH-RESULT-FOUND TRANSACTION.

READ-THROUGH-MSG.

NUMBER OF FACILITIES		LEVEL	VISIT-RATIO	SERVICE-TIME	VS-PRODUCT	CHAIN-TYPE
1	LBUS	1	.21000	100.000	21.0	1
1	GC	1	.21000	100.000	21.0	1
1	GBUS	1	.21000	100.000	21.0	1
1	GC	2	.21000	100.000	21.0	1
1	LBUS	2	.21000	100.000	21.0	1
1	PE	2	.21000	200.000	42.0	1
1	LBUS	2	.06300	100.000	6.3	1
1	GC	2	.06300	100.000	6.3	1
1	GBUS	2	.06300	100.000	6.3	1
1	GC	3	.06300	100.000	6.3	1
1	LBUS	3	.06300	100.000	6.3	1
1	PE	3	.06300	200.000	12.6	1

READ-THROUGH-RESULTS FOUND AT LEVEL 1

NUMBER OF FACILITIES		LEVEL	VISIT-RATIO	SERVICE-TIME	VS-PRODUCT	CHAIN-TYPE
1	PE	1	.49000	100.000	49.0	1

READ-THROUGH-RESULTS FOUND AT LEVEL 2

NUMBER OF FACILITIES		LEVEL	VISIT-RATIO	SERVICE-TIME	VS-PRODUCT	CHAIN-TYPE
1	LBUS	2	.14700	100.000	14.7	1
2	LSS	2	.14700	1000.000	73.5	1
1	LBUS	2	.14700	100.000	14.7	1

1	GC	2	.14700	100.000	14.7	1
1	GBUS	1	.14700	100.000	14.7	1

TAKE CARE OF LEVEL 1 UP TO LEVEL 1 BROADCAST.

NUMBER OF FACILITIES		LEVEL	VISIT-RATIO	SERVICE-TIME	VS-PRODUCT	CHAIN-TYPE
1	GC	1	.14700	100.000	14.7	1
1	LBUS	1	.14700	100.000	14.7	1
1	PE	1	.14700	100.000	14.7	1

OVERFLOW FROM LEVEL 2 BROADCAST.

NUMBER OF FACILITIES		LEVEL	VISIT-RATIO	SERVICE-TIME	VS-PRODUCT	CHAIN-TYPE
1	LBUS	1	.07350	100.000	7.4	2
1	GC	1	.07350	100.000	7.4	2
1	GBUS	1	.07350	100.000	7.4	2
1	GC	2	.07350	100.000	7.4	2
1	LBUS	2	.07350	100.000	7.4	2
1	PE	2	.07350	200.000	14.7	2

READ-THROUGH-RESULTS FOUND AT LEVEL 3

NUMBER OF FACILITIES		LEVEL	VISIT-RATIO	SERVICE-TIME	VS-PRODUCT	CHAIN-TYPE
1	LBUS	3	.06300	100.000	6.3	1
2	LSS	3	.06300	2000.000	63.0	1
1	LBUS	3	.06300	800.000	50.4	1
1	GC	3	.06300	100.000	6.3	1
1	GBUS	2	.06300	800.000	50.4	1

TAKE CARE OF LEVEL 1 UP TO LEVEL 2 BROADCAST.

NUMBER OF FACILITIES	LEVEL	VISIT-RATIO	SERVICE-TIME	VS-PRODUCT	CHAIN-TYPE

1	GC	1	.06300	100.000	6.3	1
1	LBUS	1	.06300	100.000	6.3	1
1	PE	1	.06300	100.000	6.3	1
1	GC	2	.06300	100.000	6.3	2
1	LBUS	2	.06300	800.000	50.4	2
1	PE	2	.06300	200.000	12.6	2
1	LBUS	2	.06300	800.000	50.4	2
2	LSS	2	.06300	1000.000	31.5	2

OVERFLOW FROM LEVEL 3 BROADCAST.

NUMBER OF FACILITIES		LEVEL	VISIT-RATIO	SERVICE-TIME	VS-PRODUCT	CHAIN-TYPE
1	LBUS	1	.03150	100.000	3.2	2
1	GC	1	.03150	100.000	3.2	2
1	GBUS	1	.03150	100.000	3.2	2
1	GC	2	.03150	100.000	3.2	2
1	LBUS	2	.03150	100.000	3.2	2
1	PE	2	.03150	200.000	6.3	2
1	LBUS	2	.03150	100.000	3.2	2
1	GC	2	.03150	100.000	3.2	2
1	GBUS	2	.03150	100.000	3.2	2
1	GC	3	.03150	100.000	3.2	2
1	LBUS	3	.03150	100.000	3.2	2
1	PE	3	.03150	200.000	6.3	2

STB TRANSACTION.

NUMBER OF FACILITIES		LEVEL	VISIT-RATIO	SERVICE-TIME	VS-PRODUCT	CHAIN-TYPE
1	PE	1	.30000	100.000	30.0	1
1	LBUS	1	.30000	100.000	30.0	2
1	GC	1	.30000	100.000	30.0	2
1	GBUS	1	.30000	100.000	30.0	2
1	GC	2	.30000	100.000	30.0	2
1	LBUS	2	.30000	100.000	30.0	2
1	PE	2	.30000	200.000	60.0	2
1	LBUS	2	.30000	100.000	30.0	2
2	LSS	2	.30000	1000.000	150.0	2
1	LBUS	2	.30000	800.000	240.0	2
1	GC	2	.30000	100.000	30.0	2
1	GBUS	2	.30000	800.000	240.0	2

1	GC	3	.30000	100.000	30.0	2
1	LBUS	3	.30000	800.000	240.0	2
1	PE	3	.30000	200.000	60.0	2
1	LBUS	3	.30000	800.000	240.0	2
2	LSS	3	.30000	2000.000	300.0	2

ACK TRANSACTION.

NUMBER OF	FACILITIES	LEVEL	VISIT-RATIO	SERVICE-TIME	VS-PRODUCT	CHAIN-TYPE
1	LBUS	2	.30000	100.000	30.0	2
1	GC	2	.30000	100.000	30.0	2
1	GBUS	2	.30000	100.000	30.0	2
1	GC	1	.30000	100.000	30.0	2
1	LBUS	1	.30000	100.000	30.0	2
1	PE	1	.30000	200.000	60.0	2
1	LBUS	2	.30000	100.000	30.0	2
1	GC	2	.30000	100.000	30.0	2
1	GBUS	2	.30000	100.000	30.0	2
1	GC	1	.30000	100.000	30.0	2
1	LBUS	1	.30000	100.000	30.0	2
1	PE	1	.30000	200.000	60.0	2
1	LBUS	3	.30000	100.000	30.0	2
1	GC	3	.30000	100.000	30.0	2
1	GBUS	3	.30000	100.000	30.0	2
1	GC	2	.30000	100.000	30.0	2
1	LBUS	2	.30000	100.000	30.0	2
1	PE	2	.30000	200.000	60.0	2

MAX UNBALANCED CHAIN THROUGHPUT: .001948747929455
THE CLOSED CHAIN THROUGHPUT IS: .004166652545496

CLOSED THROUGHPUT > MAX UNBALANCED THROUGHPUT BUT V9(1,10) EQUALS TO 63.00000000001 (>0) FOR THE CLOSED CHAIN, => THE SOLUTION EXISTS.

ADJUST PAPER IF NECESSARY; TYPE YES WHEN READY!
YES

```
                 QQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQ
                                        Q
                                        Q
                                  ------------
                                 |            |
 92.40000 V1                     |   240 V1   |
  373.65 V2                      |   120 V2   |
                                 |            |
```

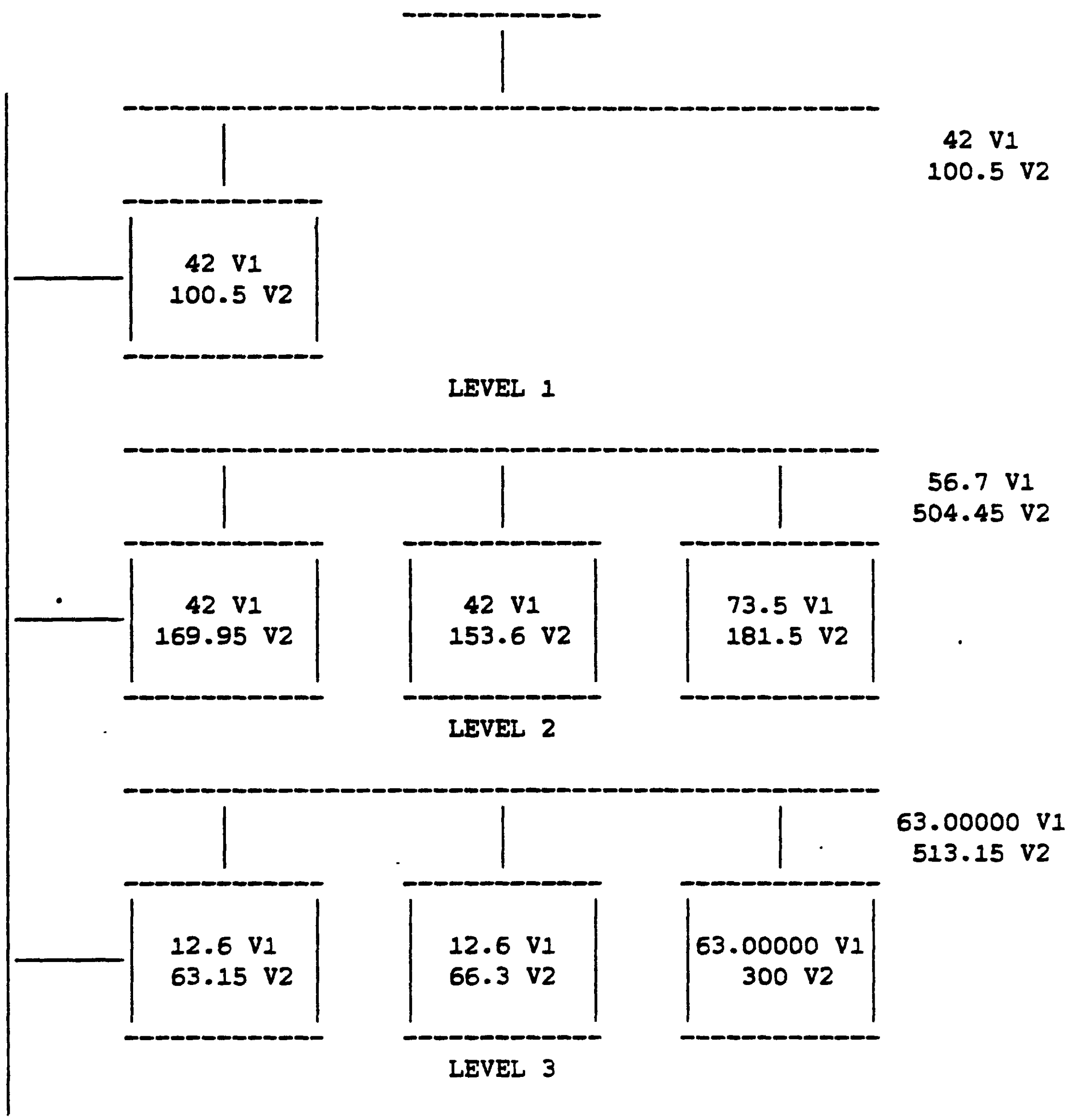


FIG-2: SUM OF (VISIT RATIO)*(SERVICE TIME) -- 1(MAIN CHAIN), 2(UAP CHAIN)

(LOCALITY,READ%)=(.7,.7), => (SYSTEM-THROUGHPUT,SYSTEM RESPONSE TIME)=(
0.001734621281393,11529.8942856).

END OF SESSION!

DO YOU WANT TO CONTINUE? CONFIRM YES/NO
NO

STOP!