PERFORMANCE ANALYSIS OF THREE INTERCONNECTION SCHEMES

FOR LARGE MULTI-MICROCOMPUTER SYSTEMS

APPROVED:

Yu-cheng Liu

David H. Wilkinson

Dean of Graduate School

TO MY MOTHER

# PERFORMANCE ANALYSIS OF THREE INTERCONNECTION SCHEMES FOR LARGE MULTI-MICROCOMPUTER SYSTEMS

by

KUMAR VENKATASUBRAMANIAM , B.E.

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at El Paso

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE

THE UNIVERSITY OF TEXAS AT EL PASO

August 1984

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to Dr. Yu-cheng Liu, my thesis adviser, for his guidance, encouragement and support.

I am grateful to Dr. Dan Patterson for his helpful comments and for serving on my thesis committee, to Dr. Po Wen Hu for the discussions in the initial stages, and to Dr. David Williams for serving on my thesis committee.

Thanks are also due to the staff of the Computer Center, particularly Mr. Marshall Adams and Mr. Poland Padilla, for their help with running the simulation programs.

This thesis was submitted to the committee on July 20, 1984.

## ABSTRACT

In this thesis, three interconnection networks (ring, binary cube and tree) for multi-microcomputer systems are studied. They are modeled as networks of queues and analytical results are obtained for performance measures such as mean queue length and mean time spent in the system by a message. These results are verified through simulation. The analysis provides a way to understand such interconnection structures and leads to better design of these systems.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

# CHAPTER 1

# INTRODUCTION

The objective of this work is to analyze the performance of three common types of interconnection schemes used in large multi-microcomputer systems, namely ring, binary cube and tree networks. Closed-form results obtained through mathematical analysis are extremely valuable tools in the design and operation of such networks. Once they have been validated through computer simulation of the networks, the various network parameters can be varied and performance measures can be obtained directly from the analytical results without the need for expensive simulations every time. It also becomes possible to compare the interconnection schemes in a useful way, in order to choose one of them for a specific application.

## 1-1 Rationale Behind Microcomputer Networks

There are basically three reasons that we are interested in a local network of microcomputers. One reason is to connect together a collection of personal computers and peripherals located close to each other, so that they are allowed to intercommunicate and share resources [13]. The shared resources would be in the form of hardware or

software, such as printers, disk drives, compilers, etc.

The second reason is to exploit the advantages of functionally distributed computing. Some of the machines could be dedicated to perform specific functions, such as file storage, data base management, terminal handling, and so on. This also leads to simpler software design because dedicated processors can reduce or eliminate multiprogramming [13].

The third reason, which is the motivation for this work, is that, with the cost of both powerful microcomputers and communication links decreasing, it is possible to build a network of microcomputers comparable in computing power to a main-frame computer but at a lower cost and higher reliability.

## 1-2 Multi-microcomputer Systems

The rapid advances in VLSI technology have already yielded single-chip microcomputers with a high ratio of computing power to cost. At present, indications are that this will continue to improve in the next few years. Device densities quadruple every three years and a detailed projection for 1985 describes a 32-bit computer chip of about 1,000,000 gates, including 200,000-300,000 for logic and microinstructions, and the rest for a read-write memory [14].

There has been considerable interest in building powerful computer systems by interconnecting hundreds or thousands of such VLSI chips. The primary advantages [15] of such systems will be reliability and cost effectiveness. The secondary advantages will be ease of growth, ease of interface and low maintenance cost.

A multi-microcomputer system does not have any shared memory. It is a loosely coupled system which provides for distributed processing. System resources are distributed over the system nodes (microcomputers).

A typical functional organization for each network node [14] is shown in Fig. 1.1. Each node is a complete microcomputer which might be fabricated on a single chip. It has two processors, one for control and I/O operations, and the other for arithmetic tasks. They share a large block of memory for arithmetic tasks, input/output operations and data buffers. The I/O processor can handle network communication functions, such as packet relaying, without degrading the local task performance significantly. The node also has ports connected to high-speed buses for communication and one port for direct memory access by peripherals and other processors.

The system nodes could be interconnected by one of several interconnection structures. The interconnection structure of a system may be defined as a graph whose nodes represent components (microcomputers) and whose edges (links)

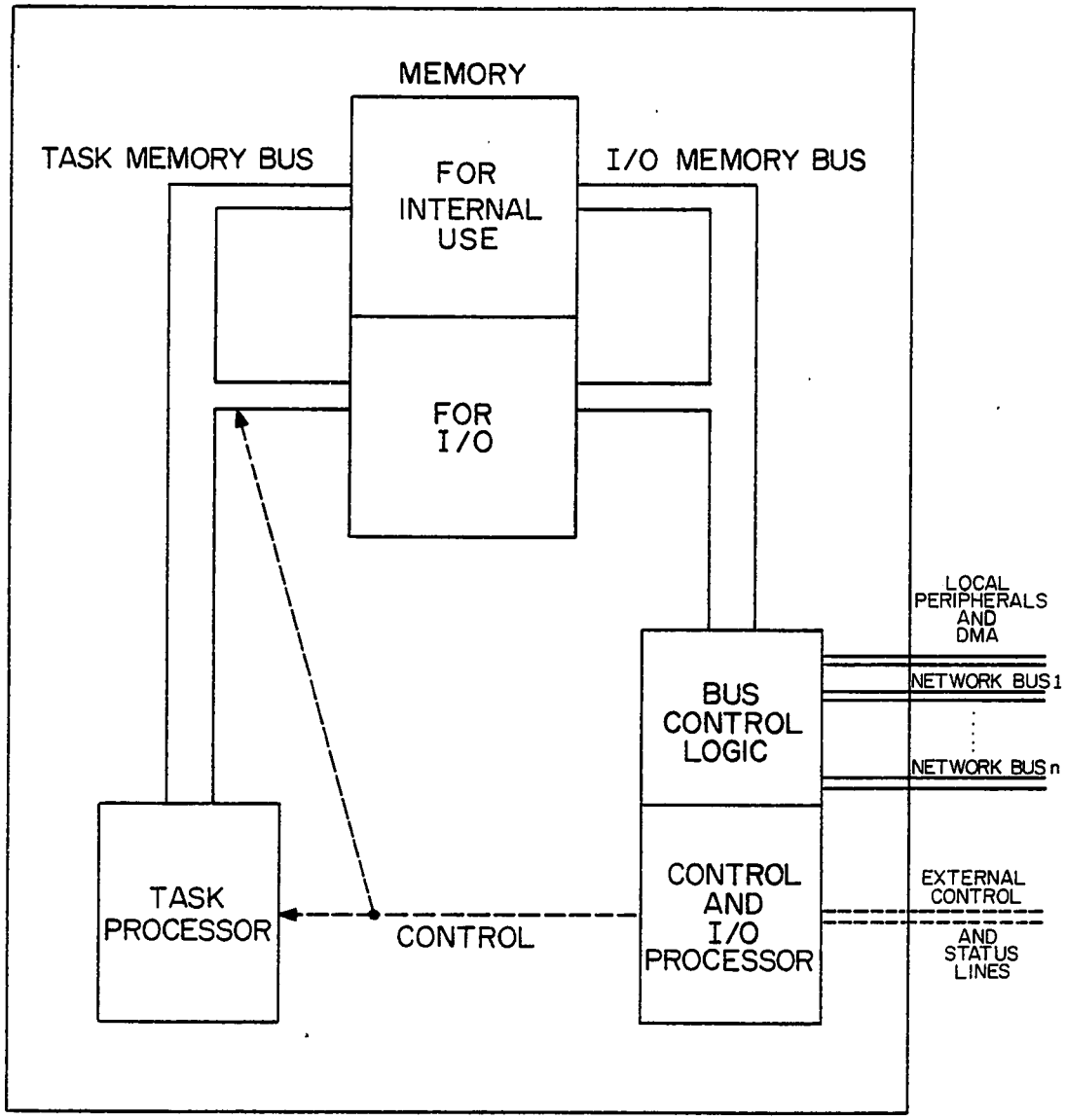



Figure 1.1 Fuctional Organization of a Single-chip Microcomputer

represent physical communication paths such as buses [6]. The links could be either dedicated to a pair of devices or time-shared by more than two devices.

Figure 1.2 illustrates two nodes connected by a system arbitration and communication bus, which is a generalized interconnection scheme [11]. This bus could be as simple as a basic master-slave control bus of about half-dozen lines, or as complex as a network of buses (links) and intermediate processors.

In a network with hundreds of microcomputers, it is clear that the nodes should be similarly constructed, particularly with respect to the number and type of bus connections, to reduce cost and complexity. Such a network or graph having the same number of connections (degree) at each node is called regular [1]. This, incidentally, simplifies the mathematical analysis considerably. The network should also be easily extensible by one or a small group of nodes at a time. This lengthens the system life-cycle by allowing for easy expansion when future applications demand it.

Another characteristic of a network is the diameter, defined as the maximum of the minimum distance between all pairs of nodes measured in number of links [1].

A good interconnection structure in general should have a small degree, a small diameter and a large number of alternate paths between a pair of nodes for fault tolerance

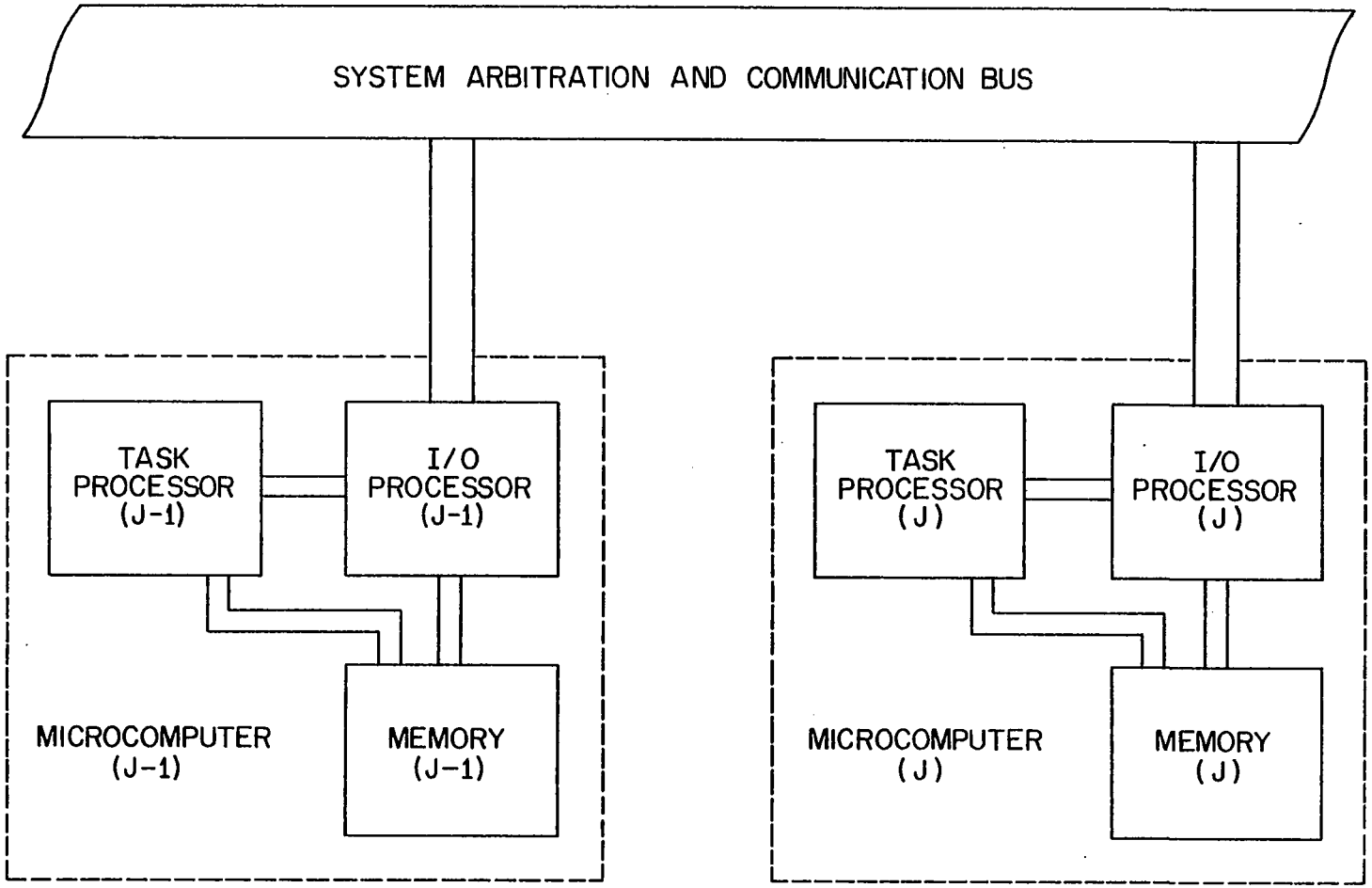



Figure 1.2 A Generalized Multicomputer Segment

[2]. In Section 1-4, we will consider the degree and diameter of the networks under study. The analysis in Chapter 2 and the simulations in Chapter 3 (for the performance measures) assume that the shortest route is always used. However, we will briefly consider the question of alternate paths in Section 1-4.

Communication through the links could be either synchronous or asynchronous [6]. Synchronous communication has the disadvantage that the time slots used for information transfer are determined by the slowest devices in the system. A popular choice for local bus communication is asynchronous communication, in which each item being transferred is preceded by a control signal to the next node on the path and the receiving node responds with an acknowledgment after receipt of the item. Thus each device in the system can operate at its own speed. This flexibility is achieved at the cost of more complex bus control logic. We will assume such asynchronous communication in our analysis, because longer messages take more time for transmission.

Each computer in the network could have its own operating system or a network operating system could be used with a portion of the operating system executing on each computer.

All the interconnection networks work in the following way: When a node creates a message, it includes the address

of the destination node in the message header. It then looks into its routing table which indicates the next node on the shortest route to the destination. The message is then added to the end of the appropriate queue (at the source) to reach the next node. A control signal precedes the actual transmission of the message. If a positive acknowledgment is not received within a certain time after transmission, the message is retransmitted. The intermediate nodes store the message in their queue buffers and forward it to the next node on the shortest path to the destination. The message need not enter a queue at the destination but instead is written into the local memory of that node.

In the event that there are several equally short paths, one of them is chosen with equal probability. If a node or link fails, the best alternate path would be used to reach the destination.

## 1-3 Survey of Published Work

In this section we survey the published material documenting research on multi-microcomputer systems in the past few years, and point out how this work fits into that scenario.

Most of the work done in the 1970's involved multiprocessor systems with shared memory. The late 1970's,

however, witnessed the beginning of a new era as microprocessors became more powerful and inexpensive. This led to the idea of building a network of microcomputers for distributed processing. One of the first papers published was by Spetz [11]. He described in a qualitative manner how hierarchical, ring, star and distributed systems perform in comparison to each other.

Sullivan and Bashkow [12] described the Columbia Homogenous Parallel Processor, an MIMD machine supporting a fully distributed hostless operating system. They considered various interconnection structures such as rings, crossbars and hierarchies briefly and modeled the boolean n-cube as a queueing network with each node as an M/D/1 model. They showed how the maximum throughput could be estimated assuming large queue lengths.

Finkel and Solomon [4] investigated the bus load, routing algorithms and the relation between average interprocessor distance and the size of the network, for four families of topologies: binary cube, snowflake, dense snowflake and star. They developed a technique for recursively constructing large networks from smaller ones, and used the recursive structure to calculate the properties of various topologies.

Wu and Liu [15] presented three cluster structures - hypercube, hierarchy and tree - as interconnection schemes

for large multi-microcomputer systems. They analyzed communication problems such as traffic congestion and message delay in these networks, as well as structural properties such as complexity, capacity and limitation. They showed how interconnection limitation could be minimized by topological optimization.

Wittie [14] introduced a dual-bus hypercube as a cost-effective method of connecting thousands of dual-port single-chip microcomputers. This structure has only two bus connections per node.

Chu, Fayolle and Hibbits [3] analyzed flow control in a tandem queueing system to reduce traffic congestion in computer networks. In such a system, whenever the queue length or the traffic intensity reaches a threshold level, input arrivals are temporarily rejected to avoid further congestion.

Lam and Lien [9] used simulation to investigate conditions for packet networks to operate at a high throughput rate. They studied the impact of network protocols for flow control, congestion control and routing on the network throughput rate. One of their results was that a uniform assignment strategy for input buffer limits is significantly better than designing them proportional to nodal traffic levels.

Arden and Lee [1] proposed a family of graphs called

multitree structured, which has the desirable properties of short internode distance, low degree and systematic structure, and is incrementally extensible.

Very recently, Bhuyan and Agrawal [2] proposed a general class of hypercube structures where the interconnection is based on a mixed-radix number system. The technique results in a variety of hypercube structures for a given number of processors, depending on the desired diameter.

The present study belongs to the same general category that the papers described above belong to. More specifically, it is an extension of the work done by Sullivan and Bashkow [12], and Wu and Liu [15]. Sullivan and Bashkow modeled each node of a binary cube as an M/D/1 queueing system. They assumed a Poisson arrival rate and a deterministic service rate. However, it seems easier to model each node as several M/M/1 queues, one for each out-going link, because the links are the resources for contention. Also, an exponential service rate is a more reasonable assumption, given the varying lengths of message packets. This simplifies the analysis to some extent. Wu and Liu derived analytical results for message delay and bus load for hypercubes, hierarchies and trees. In this work, the analysis for the tree is expanded and it is modeled as a network of queues. In addition, the ring structure is investigated and modeled also as a network of queues.

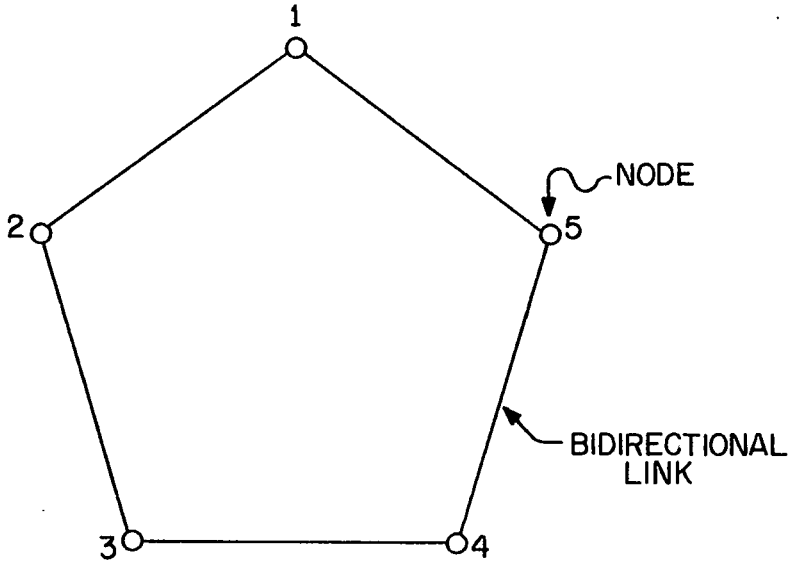
## 1-4 Interconnection Structures

Examples of the interconnection structures chosen for analysis and comparison are shown in Fig. 1.3 (ring), Fig. 1.4 (binary cube) and Fig. 1.5 (tree).

Let us now examine each of the three interconnection schemes that are being analyzed, and obtain some comparative performance measures in a qualitative manner.

The ring structure is easy to implement, particularly for large systems. Each node has only two bidirectional links connected to it. Thus the degree is only 2. Since the shortest route is always taken, the diameter is approximately N/2 for N nodes. This could become quite large as N increases. Hence, the time spent in the system by a random message is likely to be rather high. Also, since each node is likely to use many other nodes on the path to a destination, the average queue lengths would be considerable. The fault tolerance is low because if one node or link fails, two nodes on either side of the failure must go around almost the entire network to communicate with each other. This would increase traffic congestion. If there are two failures, the ring is divided into two parts with no connection between them.

The binary cube is a more complicated structure but offers more flexibility. The degree is d, which is also the



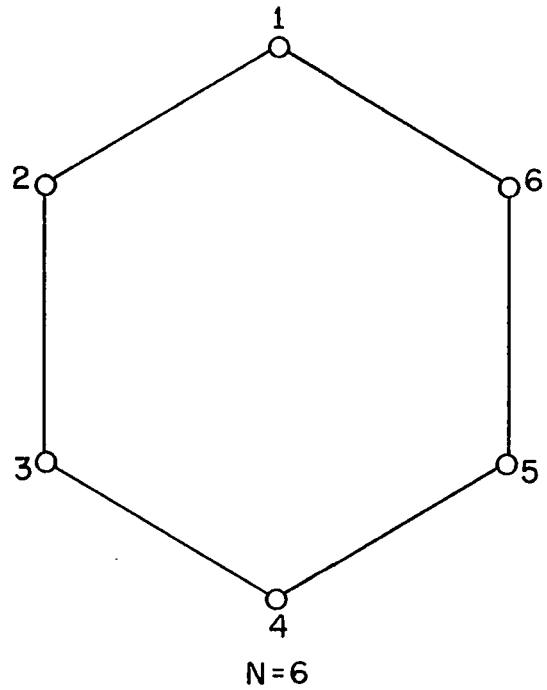




Figure 1.3 Ring Networks

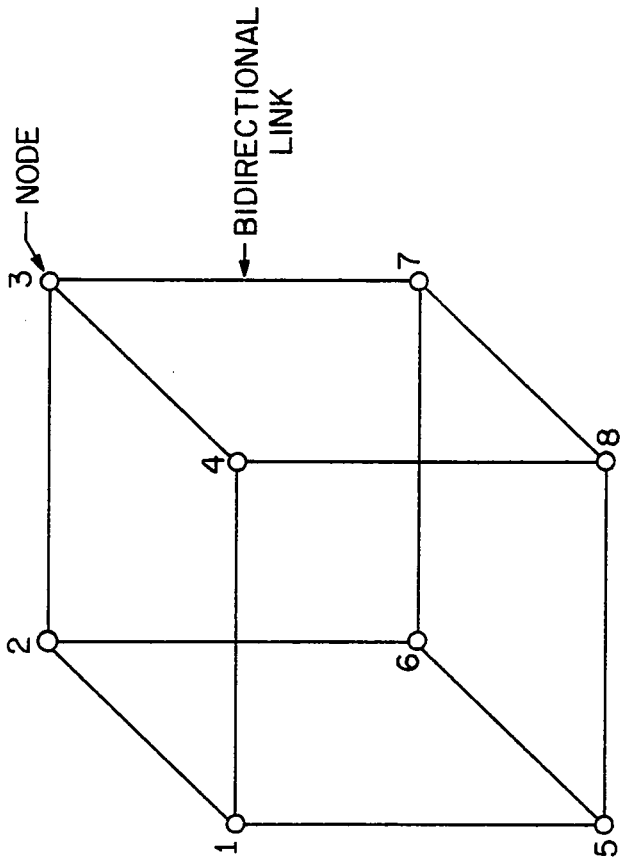



Figure 1.4 Binary Cube Network ( d = 3 )

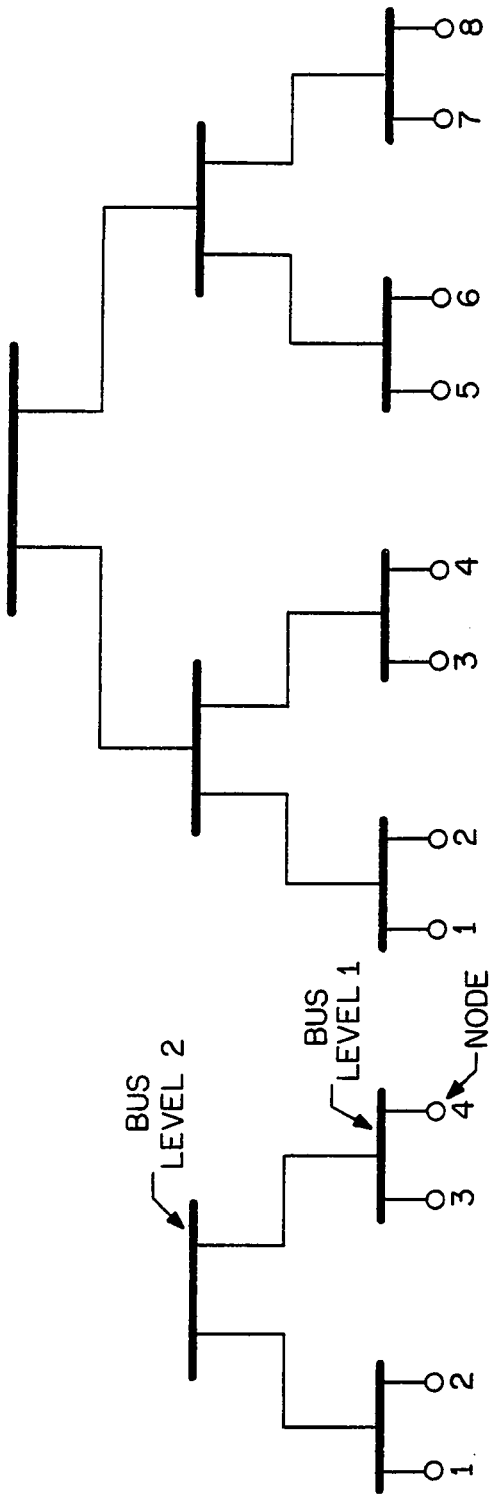



Figure 1.5 Tree Networks

dimension of the cube. In comparison with the ring, the main disadvantage is that the degree increases for each minimal expansion (to the next dimension) of the cube. On the other hand, the diameter, also equal to the dimension, is much smaller than that of an equivalent ring. Thus, the time spent in the system by a random message would be considerably smaller. It also follows that the mean queue lengths would be smaller, because each node does not use as many intermediate nodes as in a ring and consequently the traffic density is lower. Fault tolerance is excellent because of the many alternate paths available, particularly as d increases.

The tree network is conceptually and physically quite different from the other two. However, for comparison purposes, we can say that the degree is 1 regardless of the size of the tree. The diameter is difficult to estimate without mathematical analysis such as the results in Chapter 2. Intuitively, the queue lengths at higher-level buses would be higher than at lower-level buses, because there is more message traffic at higher-levels. Fault tolerance is poorer than for the ring or the cube. A single failure would divide the tree into two separate parts.

In Chapter 2, after a brief review of queueing theory, Jackson's model for networks of queues and the assumptions used, we derive analytical results for the performance measures of interest. The derivations fully rely on

Jackson's result [5,7] which allows node-by-node decomposition of a complex network.

Chapter 3 deals with the simulations performed as part of this research. The chapter begins with a description of the SLAM simulator [10] and the way programs are written in SLAM. Then, the simulation results are summarized for the networks. The independence assumption is verified through simulation and effects of varying certain simulation paramaters are studied.

Chapter 4 compares the analytical results of Chapter 2 with the simulation results of Chapter 3. It is found that the two are quite close. Suggestions are made in Chapter 5 for future research in this exciting area.

# CHAPTER 2

# ANALYTICAL RESULTS

This chapter focuses on queueing theory and its applications to the performance analysis of the three types of multi-microcomputer networks that we are studying: ring, binary cube and tree.

We begin with derivations of elementary results in queueing theory which will be useful in later analysis. Then, Jackson's model for networks of queues is described in detail. This model is the basis for all the results obtained in this chapter.

Using Jackson's model, multi-microcomputer networks are modeled as networks of queues and closed-form solutions are obtained for performance measures such as mean queue length and mean time spent in the system by a message.

## 2-1 Review of Queueing Theory

The performance of a simple M/M/1 queueing system can be measured by two parameters, the mean queue length (QL) and the mean waiting time in the system including service time (WT) [5,13]. For an M/M/1 system, the arrival rate of customers (messages) is given by a Poisson probability distribution:

$$P(n,t) = (\lambda t)^n e^{-\lambda t} / n! \qquad (2.1)$$

where

$P(n,t)$ = probability of n arrivals in time t,

and $\lambda$ = mean arrival rate.

The service time (of the single server) is an exponentially distributed random variable with a mean of $1/\mu$. The density function for the time to departure of the next customer when the system is occupied is given by

$$s(t) = \mu e^{-\mu t} \qquad (2.2)$$

The state of an M/M/1 queueing system is completely described by the number of customers currently in the system, including both queue and server. It is not necessary to describe how long the customer has been in the system, because the exponential density function has no memory.

Let p(k) be the equilibrium probability that there are k customers in the system (state of the system is k). Once these probabilities have been derived, statistical properties of the system such as $Q_L$ and $E_T$ can be found. Before proceeding further, it is necessary to make one further assumption: Transitions occur only between adjacent states during a small interval of time $\Delta t$; i.e., the probability of

more than one arrival or of more than one service or of one arrival and one service together in $\Delta t$ is zero. Such systems are known as birth-death systems.

The equilibrium state probabilities will be derived using the principle of detailed balancing. Referring to Fig. 2.1, it is clear that transitions from state k to state k+1 occur at the rate of $\lambda\, p(k)$, because p(k) is the probability that the system is in state k, $\lambda$ is the arrival rate and transitions are assumed to occur only between adjacent states. Similarly, transitions from state k to state k-1 occur at the rate of $\mu\, p(k)$, where $\mu$ is the service rate. At equilibrium, the rate at which transitions occur from state k to state k+1 must be balanced by the rate at which transitions occur from state k+1 to k.

Therefore,

$$\lambda\, p(0) = \mu\, p(1) \qquad (2.3)$$

$$\lambda\, p(1) = \mu\, p(2) \qquad (2.4)$$

In general,

$$\lambda\, p(k) = \mu\, p(k+1) \qquad (2.5)$$

and the general solution is

$$p(k) = \rho^k\, p(0) \qquad (2.6)$$

where $\rho = \lambda/\mu$. This is known as the traffic intensity.

To eliminate p(0), the sum of the state probabilities can be set to 1:

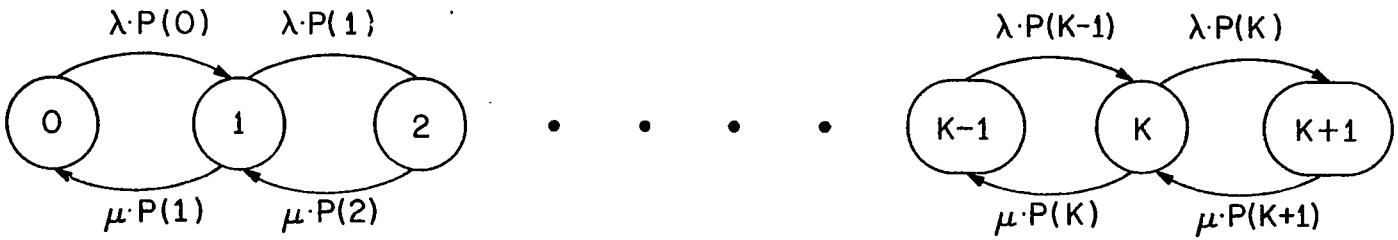



Figure 2.1 State Diagram for M/M/1 Queueing System

$$\sum_{k=0}^{\infty} \rho^k \cdot p(0) = 1 \qquad (2.7)$$

Since $\sum_{k=0}^{\infty} \rho^k = 1/(1-\rho)$ (sum of geometric series),

we find that $p(0) = 1-\rho$ and therefore

$$p(k) = (1-\rho)\rho^k \qquad (2.8)$$

The mean number of customers in the system, N, can be found as follows:

$$N = \sum_{k=0}^{\infty} k\, p(k) = \rho/(1-\rho) \qquad (2.9)$$

Using Little's result, the mean waiting time, including service time , is:

$$WT = \frac{N}{\lambda} = \frac{1}{\mu - \lambda} \qquad (2.10)$$

The mean queue length can be obtained by subtracting the expected number in service from the expected number in the system, N. The probability that the server is busy is 1 - p(0). Therefore, the expected number in service is:

$$S = 1 - p(0) \qquad (2.11)$$

From Eq.(2.8), the mean queue length is:

$$QL = N - S = \frac{\rho}{1-\rho} - \rho$$

$$= \frac{\lambda^2}{\mu(\mu-\lambda)} \qquad (2.12)$$

These results have been derived in detail elsewhere [5,13].

## 2-2 Networks of Queues

The networks that are considered will have the following characteristics [5,7]:

1. The networks contain more than one service center.
2. Each service center is a multi-channel queue in general, with each channel having an identical exponential service time distribution.
3. Arrivals at any center may come from outside the system or from other service centers in the network.
4. Arrivals from outside occur as a Poisson distribution.
5. When a unit completes service at a particular center, it may leave the system or be routed to another center.
6. There is unlimited waiting space at every service center.
7. Total arrival rate at every center is less than its potential service rate.

Such networks are known as Jackson networks [7]. Jackscn's main result is that if one properly defines the mean arrival rate at the various service centers then the steady-state distributions at those centers look exactly like standard multi-channel systems. This means that it is possible to decompose a complex network into a number of simpler subsystems, as explained below.

Node-by-node decomposition in a network can be justified on the basis of three important properties of Poisson/exponential streams:

1. Combining a finite number of independent Poisson streams yields a Poisson stream whose mean is equal to the sum of the means of the component streams.

2. Splitting a Poisson stream of rate $\lambda$ in random fashion, with portion r(i) going to stream i yields a set of new streams which are Poisson with rates $r(i) \cdot \lambda$ .

3. Passing a Poisson stream through an exponential service facility yields a departure stream which is Poisson with the same parameter as the input stream.

Figure 2.2 illustrates the first and third properties.

By using these properties, we can compute the effective arrival rate from all sources for each service center in the network and then simply treat each center as a separate M/M/1 or M/M/c subsystem. This greatly simplifies the analysis of complex networks and all results available for simple

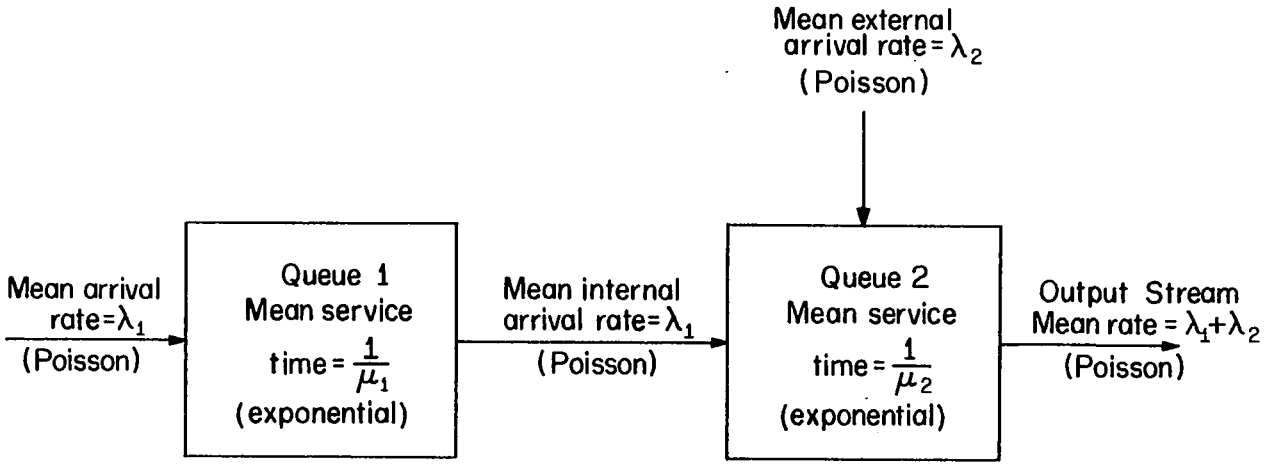



Figure 2.2 Illustration for Jackson's Networks.

queueing systems are applicable. In steady state, the total flow rate into node i can be expressed as

$$e = \lambda_i + \sum_{k=0}^{N} r_{ki}\, e_k \quad , k \neq i \qquad (2.13)$$

where

$\lambda_i$ = arrival rate due to external cause,

$r_{ki}$ = probability that upon completing service at node k customers will go to node i,

e = effective arrival rate at node i from all sources.

These results will be utilized in the last three sections of this chapter, in the derivation of analytical results for the performance measures of multi-microcomputer networks.

## 2-3 The Independence Assumption

Before proceeding further, it is necessary to consider the relationship between interarrival times and the lengths of adjacent messages [8]. For any particular node in the network, the lengths of messages generated and the interarrival times are indeed dependent. However, the collective interarrival times and lengths of messages

generated by the entire group of nodes do exhibit some degree of independence. The reason for this is that the mutiplicity of paths in and out of a node considerably reduces the dependency between interarrival times and message lengths as they enter various queues in the network. Such an assumption of independence is necessary for our analysis.

In Chapter 3, the simulation results demonstrate that this is a reasonable assumption to make. For the present, we will simply assume that each time a node receives a message, it chooses a new length for it (v) from the probability density function $p(v) = \mu e^{-\mu v}$, and then forwards it to the next node.

## 2-4 General Assumptions

It is necessary at this point to list all the general assumptions regarding the functional aspects of the three types of networks. Any assumptions specific to a particular configuration will be included in that section. The general assumptions under which analytical models are developed, are as follows:

1. Each node generates messages in Poisson fashion with a mean of $\lambda$ messages/unit time.
2. Each node tries to communicate with all other nodes with equal probability.

3. The service time (by the out-going links at the source or intermediate nodes) to transmit messages is exponentially distributed with a mean of $1/\mu$ time units. This ensures that the departing stream is also Poisson.

4. Each message has the address of the destination in its header. The source node and all intermediate nodes always route the messages to the next node on the shortest path to the destination, by using the routing tables stored in memory.

5. Each message is handled by all nodes in a store-and-forward fashion. The queueing delay at each node is only due to the limited speed of the links (servers). There is no queueing delay at the destination. Queueing is required only if an out-going link is needed for transmission.

6. There are no equipment failures and hence there is no need to consider alternate paths or retransmission of the message, although these would be built into real systems for fault tolerance.

7. The queue buffer for each queue in the system has unlimited capacity. (In a practical system, the buffer will have finite capacity and the sending node will wait until sufficient space is available in the receiving node).

## 2-5 Ring Networks

Figure 1.3 shows typical ring networks with odd and even number of nodes. The nodes are connected by high-speed bidirectional links. Such networks can be modeled as networks of queues. Each node can be modeled as two independent M/M/1 queues, one for each direction. The two links carrying messages out of a node are modeled as two queues, because they are the resources for contention.

Based on the discussion in Section 2-2, we can perform a node-by-node decomposition of the network using the three important properties of Poisson/exponential streams. This allows us to consider the subsystems (nodes) rather than the entire network. The important thing is to accurately estimate the input arrival rate into a queue. Once this is done, results from Section 2-1 can be applied to obtain closed-form expressions for the mean queue length and the mean time spent in the system by a message.

The specific assumptions needed to analyze this configuration are:

1. Each node has two buffers to store the waiting messages, one for each direction.

2. A message is transmitted by the source in either the clockwise or the counter-clockwise direction, depending on whichever leads to the shortest path. For a network with an

even number of nodes, the farthest destination for any source can be reached by either direction. Then, one direction will be chosen with a probability of 0.5.

We will first derive an expression for the input arrival rate in one direction for any node in a network with an odd number of nodes. By symmetry, the same result will hold for all nodes in the network (for both directions).

Let each node generate $\lambda$ messages/unit time. Since the shortest route is always taken, $\lambda/2$ messages/unit time will be transmitted in either direction. Consider Fig. 1.3(a) which shows 5 nodes. At node 1, in the clockwise direction, nodes 1, 2 and 3 are the only possible sources. However, node 3 can not send through node 1 because to reach node 5, it would transmit in the counter-clockwise direction (for the shortest route). Therefore, we need to consider only nodes 1 and 2 in this case.

Node 2 sends to nodes 1 and 5 with equal probability (in the clockwise direction). If the destination is node 1, the message will not queue up at node 1; otherwise it will.

The effective input arrival rate into node 1 in the clockwise direction is:

$e$ = arrival rate due to node 1
+ arrival rate due to node 2
$= \lambda/2 + (1/2).(\lambda/2) = 0.75\lambda$

Following the same reasoning, we can generalize this for

N nodes:

$$e = \lambda/2 + (\lambda/2) \sum_{k=1}^{\text{Trunc}(N/2)-1} \frac{\text{TRUNC}(N/2)-k}{\text{TRUNC}(N/2)} \qquad (2.14)$$

The derivation of e for an even number of nodes is similar. Fig. 1.3(b) shows 6 nodes. Now we need to consider node 3 also as a possible source affecting the arrival rate at node 1. Node 3 transmits to node 6 only half as often as it does to nodes 2 or 1, since node 6 can be reached in both directions. It affects the arrival rate at node 1 only when it transmits to node 6. Also, node 2 affects the arrival rate at node 1 when it transmits to nodes 5 or 6.

As before, the effective arrival rate at node 1 is:

e = arrival rate due to node 1
+ arrival rate due to node 2
+ arrival rate due to node 3
$= \lambda/2 + (1.5/2.5)(\lambda/2) + (0.5/2.5)(\lambda/2) = 0.9\lambda$

Again, we can generalize this for N nodes:

$$e = \lambda/2 + (\lambda/2) \frac{(N/2-1) + \sum_{k=1}^{N/2-2} 2k}{(N-1)} \qquad (2.15)$$

Using the above equations, we can obtain the mean queue length and the mean waiting time:

$$QL = \frac{e^2}{\mu(\mu - e)} \qquad (2.16)$$

$$WT = \frac{1}{\mu - e} \qquad (2.17)$$

The average distance (AD) traveled by a message is obtained as follows. Let us first consider an odd number of nodes. Let

$$N = 2k + 1 \qquad (2.18)$$

k is the number of destinations in each direction for a source. Considering any one direction,

$$AD = \frac{1+2+\ldots+k}{k}$$

$$= \frac{k+1}{2} \qquad (2.19)$$

Next we consider an even number of nodes. The farthest destination for a source is at a distance of

$$k = N/2 \qquad (2.20)$$

There are two possible destinations (in two directions) at distances of 1, 2, ... ,k-1 hops from the source. Therefore,

$$AD = \frac{2(1+2+\ldots+k-1)+k}{2k-1}$$

$$= \frac{k^2}{2k-1} \qquad (2.21)$$

Using Eq. (2.16), the mean time spent in the system (TIS) by a message is:

TIS = (WT in each node) . (mean distance traveled)

$$= WT \cdot AD \quad \text{time units} \qquad (2.22)$$

## 2-6 Binary Cube Networks

Figure 1.4 shows a typical binary cube network of 3 dimensions. The adjective 'binary' derives from the fact that each link is connected to only two nodes regardless of the dimension of the network. For a network of dimension $d$, there are $N = 2^d$ nodes.

Each node has $d$ bidirectional links connected to it. Therefore, we can model each node as $d$ independent M/M/1 queues, following the same reasoning as in the previous section. The specific assumptions regarding the network are as follows:

1. Each node has $d$ buffers, one for each out-going link.

2. Whenever there are several equally short paths to a destination, one is chosen with uniform probability.

We will now derive an expression for the input arrival rate into any of the Nd queues in the network. By symmetry, all nodes behave the same way.

First, we must find the average distance traveled by a random message. Consider a cube with dimension d. A node can be represented as a binary d-dimensional vector. For any given d-dimensional vector, there are $\binom{d}{i}$ vectors that have i positions different from the given vector. Shortest paths are indicated by the queue numbers corresponding to the positions where the two vectors differ. Therefore, each node has d nodes at a distance of one hop, $\binom{d}{2}$ nodes at a distance of two hops, and so on. There is one node at the maximum distance of d. Therefore the average distance (AD) to a destination is:

$$AD = \frac{1.\binom{d}{1}+2.\binom{d}{2}+\ldots+d.\binom{d}{d}}{\binom{d}{1}+\binom{d}{2}+\ldots+\binom{d}{d}}$$

$$= \frac{d\,2^{d-1}}{2^{d}-1} \qquad (2.23)$$

Note that AD is expressed in number of links traveled.

It is also equal to the number of queues that a message has to go through, counting the source and all intermediate nodes but excluding the destination.

Probability that a random message will use node i as its path is given by:

$$\text{Prob(i is not the destination)} \times \frac{\text{(average no. of nodes on path excluding source \& destination)}}{\text{(total no. of intermediate nodes available)}}$$

Hence,

$$P(\text{message will use node i}) = \frac{N-2}{N-1} \cdot \frac{AD-1}{N-2}$$

$$= \frac{AD-1}{N-1} \qquad (2.24)$$

For each node, there are N-1 possible sources contributing to the arrival rates, each generating $\lambda$ messages/unit time. Therefore the total rate at which other nodes send messages through a given node is:

$$\text{Total rate} = P(\text{a message will use node i}) \cdot (\text{rate of messages generated by other nodes})$$

$$= \frac{AD-1}{N-1} (N-1) \cdot \lambda$$

$$= (AD-1) \cdot \lambda \qquad (2.25)$$

Since there are d incoming lines, the arrival rate due to each of them is:

$$(1/d) \cdot (AD-1) \cdot \lambda$$

By symmetry, this is also the rate that must flow out on each of the d out-going lines at that node.

Therefore, the effective arrival rate into each of the d queues at any node is:

e = rate due to host node

+ rate due to other nodes

$$= \lambda/d + (AD-1)\,\lambda/d$$

$$= \frac{\lambda \cdot 2^{d-1}}{2^{d} - 1} \qquad (2.26)$$

As before, QL and WT are given by Eq. (2.16) and Eq. (2.17). The average distance traveled by a random message is AD from Eq. (2.23). Therefore, the mean time in the system for a message is:

$$TIS = AD \cdot WT \qquad (2.27)$$

## 2-7 Tree Networks

Figure 1.5 shows typical tree networks. L is the number of levels and K is the number of branches (links) coming out of each bus to lower level buses. The buses are connected by

high-speed,bidirectional links and the nodes are connected to the buses at the lowest level.

Such structures are modeled a little different from the other two configurations. Here the buses are the resources for contention and hence each bus is modeled as a server in an M/M/1 system. Note that the any given bus will have a higher load than each of the many links connected to it. The following specific assumptions must be made:

1. Each bus has a buffer associated with it, to store all the waiting messages.
2. Each message first enters the queue at the lowest-level bus to which the source node is connected. It then enters queues in other buses until it reaches the destination.
3. The routing table at each bus shows the next bus (or the node in case the last bus has been reached) on the shortest path to the destination.
4. All messages coming into a bus from all sources are stored in a common buffer. One message is forwarded at a time.

For a tree of L levels and K branches, the number of nodes is [15]:

$$N = K^{L} \qquad (2.28)$$

Since each node communicates with all other nodes with

uniform probability, the probability of a level i message from a node is:

$$p(i) = \frac{(K-1)\,K^{i-1}}{K^{L}-1} \qquad (2.29)$$

For example, referring to Fig.2.4(b), the probability of a level 2 message from a given node would be p(2)=2/7. This can also be obtained by inspection assuming a uniform reference model.

The number of level i messages per unit time, from a given node, is:

$$m(i) = p(i)\cdot\lambda \qquad (2.30)$$

The next step is to determine the effective input arrival rate into a bus at level i, from all sources:

e(i) = arrival rate due to level i messages
+ arrival rate due to level j messages
(i<j<=L)
= (no. of nodes in level i subsystem).m(i)
+(rate at which messages go from level i subsytem to all level j buses,through level i bus)
+(rate at which messages come into level i subsystem from higher buses, through level i bus)

By symmetry of the structure, the second and third terms are equal. Hence it is sufficient to determine only the second term and multiply it by 2 [15].

e(i) = (no. of nodes in level i subsystem).m(i)
+ 2.(no. of nodes in level i subsystem).
.(rate of level i messages from
level i subsystem)

$$= K^{i} m(i) + 2 K^{i} \sum_{j=i+1}^{L} m(j) \qquad (2.31)$$

For the special case where i=L,

$$e(L) = K^{L} m(L) \qquad (2.32)$$

Note that unlike the other networks discussed earlier, the input arrival rate is different for buses at different levels in the tree. For each level, we can once again apply Eq. (2.16) and Eq. (2.17) to obtain QL and WT.

The next step is to determine the mean time spent in the system by a random message (TIS). Let D(i) represent the total delay suffered by a level i message. Observing that a message must go upto the level i bus and come down all the way to its destination, we have for the general case:

$$D(i) = \sum_{K=1}^{i-1} 2\ WT(K) \quad + WT(i) \qquad (2.33)$$

For $i=1$, $D(1) = WT(1)$ (2.34)

Now it is easy to obtain TIS by taking the overall expected value of the delays ( D(i) ) at the various bus levels.

$$TIS = \sum_{i=1}^{L} p(i)\ D(i) \qquad (2.35)$$

Note that p(i) is the probability of a level i message, from Eq. (2.29).

# CHAPTER 3

# SIMULATION RESULTS

This chapter is devoted to the discussion of the various simulations performed to validate the analytical results obtained in Chapter 2. Actual comparison of the two will be done in Chapter 4.

The simulation language used is called SLAM, a simulation language for alternative modeling [10]. We will discuss briefly the nature of the SLAM simulator and the way programs are written in SLAM. Sample programs for each of the three types of networks are included in the Appendices. The simulation results are summarized in the last two sections of this chapter. The simulation programs were run on an IBM 4341 system.

## 3-1 The SLAM Simulator

SLAM is an advanced FORTRAN based language that allows simulation models to be built based on different 'world views', as described below.

Discrete simulation is used when the variables of the model change discretely at specified points in simulated time, called event times. In continuous simulation, the variables may change continuously over simulated time.

Discrete simulation is further classified in SLAM as event orientation, activity scanning orientation and process orientation.

In the event-oriented world view, a system is modeled by defining the changes that occur at event times. The modeler must determine the events that can change the state of the system. In the activity scanning orientation, the modeler describes the activities in which the entities in the system engage and prescribes the conditions which cause an activity to start or end. The process or network orientation provides statements, each of which actually defines a sequence of events involved in elements such as a queue or a server. These statements can be employed to model the flow of entities in a system.

Of the various approaches available in SLAM, network modeling is almost tailor-made for simulating the networks that we are studying. The network structure consists of special symbols called nodes and branches (not to be confused with nodes and links in a computer network). These symbols model elements in a process such as queues, servers and decision points. The modeling task consists of combining these symbols into a network which represents the system of interest. The entities (messages) in the system flow through the network as in a real system. The network model, written as a series of statements, is input to the

SLAM simulator. The simulator collects appropriate statistics such as mean queue lengths, mean waiting times and average server utilization during the simulation period, the length of which must be specified.

## 3-2 Structure of SLAM Programs

A network of queues is modeled in SLAM as follows. If, for example, queue number i is being modeled along with its server(s), the necessary input statements are:

QUEUE(i); (a node)

ACT(N)/i,DUR,PROB or COND,NLBL; (a branch)

Here N is the number of parallel servers, i is the activity number, DUR is the duration of the specified activity, PROB is the probability of selecting the activity, COND is the condition for selecting it (only for non-service activities), and NLBL is the label of the end node to which branching is required. All service activities (following the queues) must have N and i specified. Non-service activities for branching do not need these parameters.

Conditional or probabilistic branching could be performed for message routing after each service activity. The ACT statement is used to represent a non-service activity here. For example, if branching is required to two

labels with equal probability, we can write

```
ACT,,0.5,L1;
ACT,,0.5,L2;  (non-service activities)
```

CREATE nodes are used to create messages at a specified rate:

```
CREATE,TBC,TF,MA,MC,M;
```

Here TF is the time the first entity is created, TBC is the time between creation of successive entities, MA is the attribute number of each entity in which its creation time is stored, and MC is the maximum number of entities to be created.

The ASSIGN node is used to prescribe values to the attributes of an entity passing through it or to prescribe values to any system variables. The statement for such a node is

```
ASSIGN,VAR=VALUE,VAR=VALUE,.....,M;
```

M is the number of points to which branching will be done. ASSIGN nodes are used in our programs to assign the destination address to an attribute of each entity, which is then used as a routing condition.

The TERM node terminates entities from the system after the destination is reached. The COLCT node can be used to collect statistics such as the mean time spent in the system by an entity, in addition to the information normally printed out by SLAM.

Programs for the ring are the simplest. Each message is tagged with the number of its destination as one of its attributes. It is routed to the next system node on the path until the next system node is the destination, in which case it is terminated. (A system node is a node in the actual computer network, not a SLAM node). A decision point follows each system node after the server.

Programs for the tree networks are written as follows. Again each message has the destination number as one of its attributes. The message is first sent to the level 1 bus to which the system node is connected. If the destination is connected to the same bus, the message is sent to it directly from this bus. If not, the message goes to the next higher bus and the process is repeated. There are decision points following the queue and server at each bus.

Programs for the cube networks are the most complicated. Each system node has d queues (d is the dimension). Again, the destination address is one of the attributes of each message. In order to make sure that a message will always take the shortest path and enter the correct queue at each system node, we have devised the following method. Each system node is assigned a binary address of d bits. Each of the d queues in a system node is assigned a queue number between 1 and d. If the Hamming distance between the source and the destination is 1, then

there is only one shortest path and the message is placed in the queue number of the source corresponding to the bit position that differs in the two addresses. If the Hamming distance is greater than one, there is more than one shortest path. In this case, the queues corresponding to the paths are again indicated by the bit positions where the two addresses differ. One of these queues is chosen with uniform probability. The process is repeated at all intermediate system nodes for a particular message.

There is a routing table (decision point in the program) at each system node. Table 3.1 shows the routing table for one node in a cube with d=4.

## 3-3 Summary of Results

In all, five different ring networks, two cube networks and five tree networks were simulated. Tables 3.2, 3.3 and 3.4 summarize the results for the three types of networks respectively. In all tables, case I indicates that the message length is recomputed at each node, and case II indicates that the message length is retained without change until termination of the message. Comparison of the two cases justifies the independence assumption. For all cases, $\lambda$ =1 message/unit time and $\mu$ = 20 messages/unit time.

Table 3.1

Routing Table for a Cube with d = 4
(at node 0)

| Source Address | Destination Address | Queue Numbers at Node 0 |
|---|---|---|
| 0 0 0 0 | 0 0 0 1 | 4 |
| | 0 0 1 0 | 3 |
| | 0 0 1 1 | 3,4 |
| | 0 1 0 0 | 2 |
| | 0 1 0 1 | 2,4 |
| | 0 1 1 0 | 2,3 |
| | 0 1 1 1 | 2,3,4 |
| | 1 0 0 0 | 1 |
| | 1 0 0 1 | 1,4 |
| | 1 0 1 0 | 1,3 |
| | 1 0 1 1 | 1,3,4 |
| | 1 1 0 0 | 1,2 |
| | 1 1 0 1 | 1,2,4 |
| | 1 1 1 0 | 1,2,3 |
| | 1 1 1 1 | 1,2,3,4 |

Table 3.2

Simulation Results: Ring Network

| No. of Nodes N | Mean Queue Length (no. of messages) | | Mean Time in System (time units) | |
|---|---|---|---|---|
| | case I | case II | case I | case II |
| 5 | 0.00144 | 0.00156 | 0.07597 | 0.07605 |
| 6 | 0.00208 | 0.00220 | 0.09203 | 0.09206 |
| 8 | 0.00344 | 0.00380 | 0.1190 | 0.1200 |
| 16 | 0.01330 | 0.02182 | 0.2343 | 0.2515 |
| 20 | 0.02150 | 0.04183 | 0.2992 | 0.3397 |

Table 3.3

Simulation Results: Cube Network

| No. of Nodes N | Mean Queue Length (no. of messages) | | Mean Time in System (time units) | |
|---|---|---|---|---|
| | case I | case II | case I | case II |
| 8 | 0.000858 | 0.000875 | 0.08677 | 0.08615 |
| 16 | 0.000790 | 0.000850 | 0.10670 | 0.10730 |

Table 3.4

Simulation Results: Tree Network

| No. of Levels L | No. of Branches K | Bus Level | Mean Queue Length (no. of messages) | | Mean Time in System (time units) | |
|---|---|---|---|---|---|---|
| | | | case I | case II | case I | case II |
| 2 | 2 | 1 | 0.0319 | 0.0408 | 0.1349 | 0.1414 |
| | | 2 | 0.0198 | 0.0266 | | |
| 2 | 3 | 1 | 0.0939 | 0.1041 | 0.1726 | 0.1785 |
| | | 2 | 0.1753 | 0.1960 | | |
| 3 | 2 | 1 | 0.0416 | 0.0531 | 0.2463 | 0.2628 |
| | | 2 | 0.1156 | 0.1476 | | |
| | | 3 | 0.0658 | 0.0920 | | |
| 2 | 4 | 1 | 0.2113 | 0.2214 | 0.2616 | 0.2607 |
| | | 2 | 1.3218 | 1.2627 | | |
| 4 | 2 | 1 | 0.0481 | 0.0615 | 0.4366 | 0.4682 |
| | | 2 | 0.1870 | 0.2245 | | |
| | | 3 | 0.6298 | 0.7746 | | |
| | | 4 | 0.3220 | 0.3760 | | |

## 3-4 Effects of Varying the Simulation Parameters

We have used 10,000 time units as the duration of all the simulation runs performed in this work. The reason for choosing that figure is that the statistics collected during the simulation runs seem to converge and settle down at certain values as the simulation time increases. From Fig. 3.1 and Fig. 3.2, it is clear that at 10,000 time units the convergence has already occured. This could be thought of as the minimum time required for the simulated system to reach steady-state.

We mentioned in Section 3-3 that the independence assumption is quite reasonable because the results obtained for case I and case II are very close to each other. It is of interest to study what parameters, if any, would have an effect on the closeness of the two sets of results. Obviously, simulation time is not a candidate for this study because we are interested only in steady-state results. The only other variables are the message generation rate $\lambda$ and the service rate $\mu$. It would be sufficient to vary just one of these because, in effect, we are varying the traffic intensity.

Figure 3.3 shows the mean queue length as a function of $1/\mu$, the mean service time. Case I and case II are both

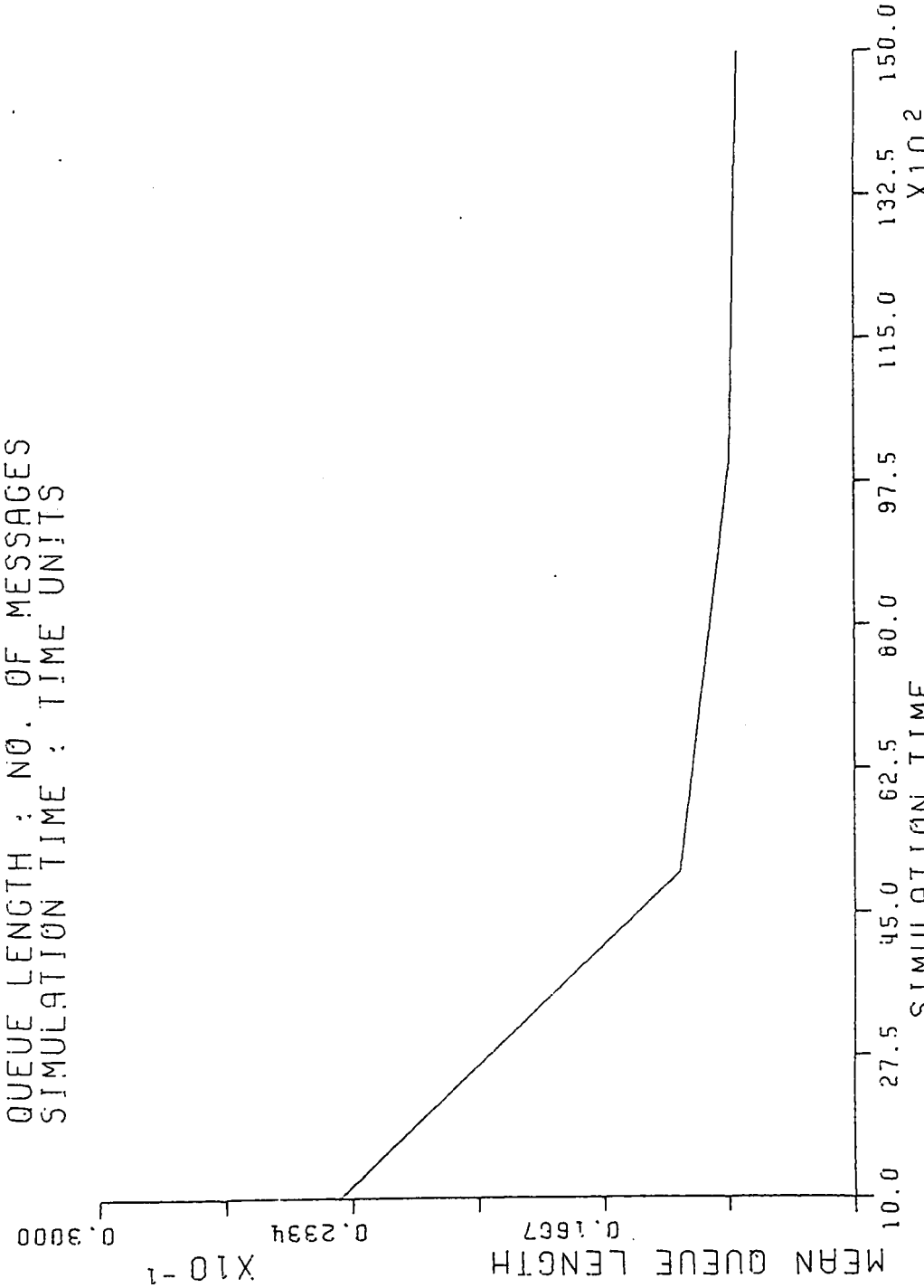



Figure 3.1. Variation of Mean Queue Length with Simulation Time

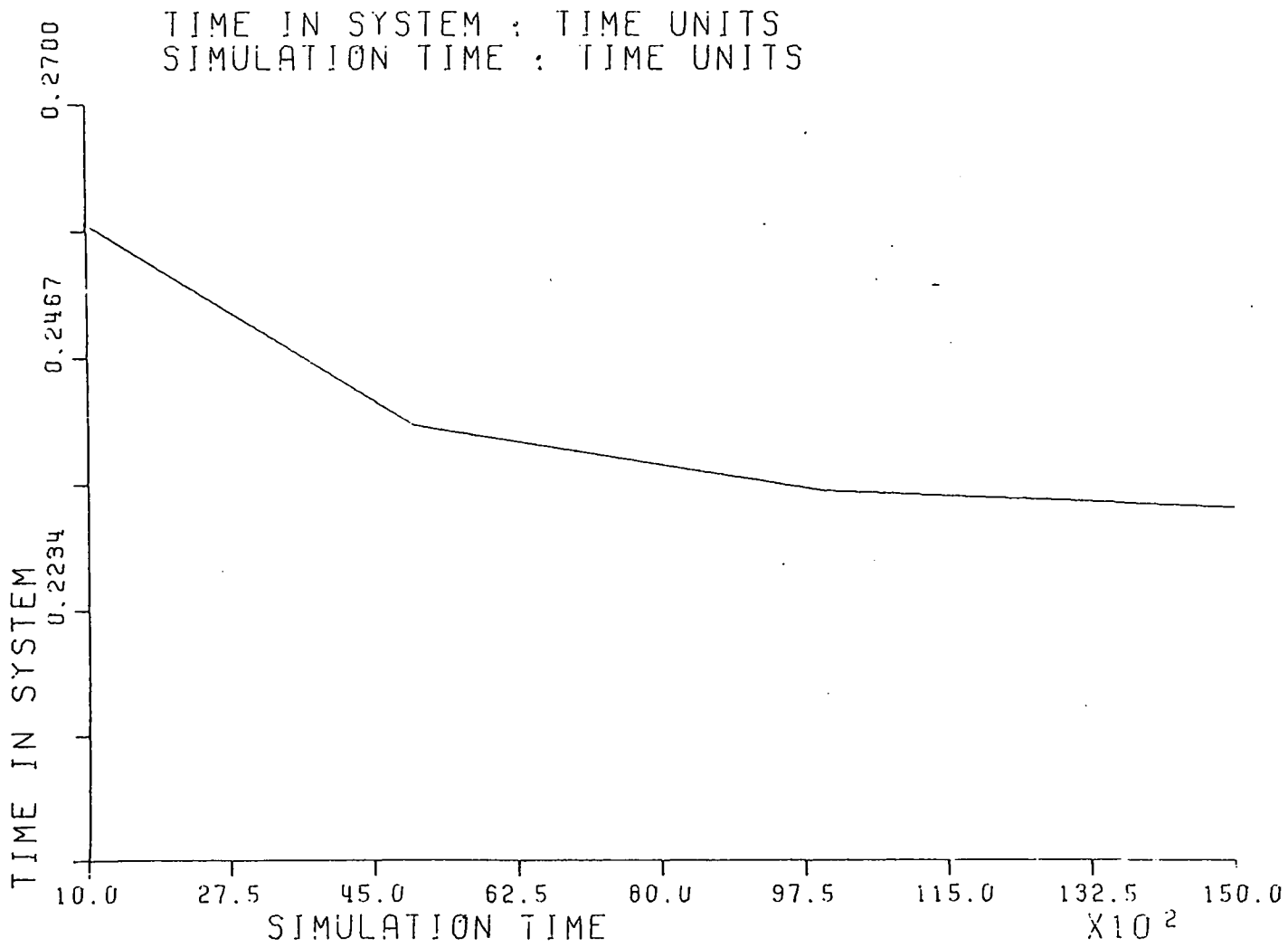



Figure 3.2. Variation of Mean Time in System with Simulation Time

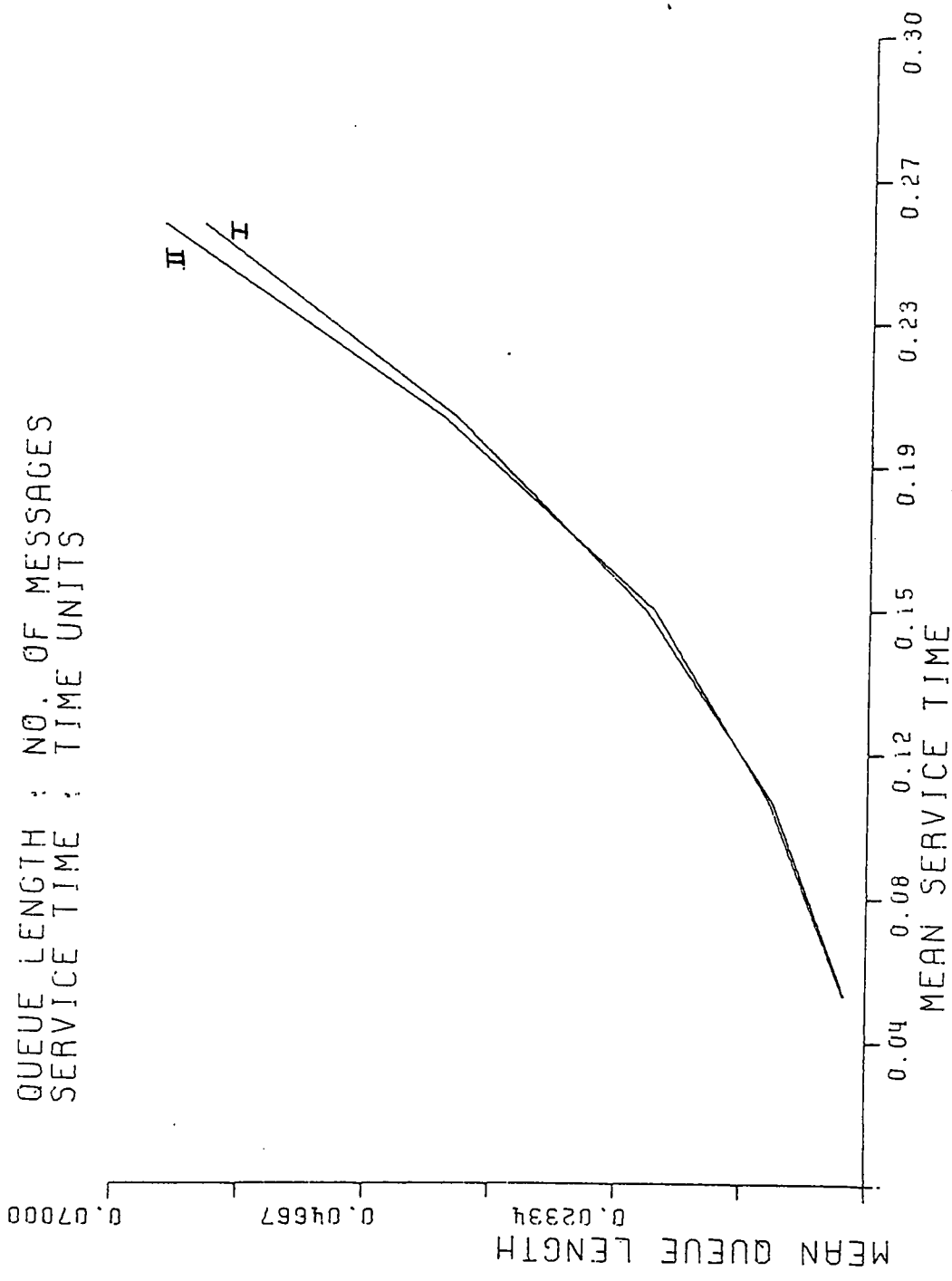



Figure 3.3. Mean Queue Length vs Mean Service Time: Case I and Case II

plotted for comparison. The difference between the two cases increases for higher service times.

## 3-5 Problems with Simulation

While performing simulation runs in SLAM, the following problems were encountered:

1. The CPU time required on an IBM 4341 system was frequently more than 30 minutes for networks with 15 to 20 system nodes, particularly for tree and cube networks. This shows that simulation of very large networks would be quite expensive. Therefore, analytical results are necessary for the design of large systems.

2. For some of the more complicated networks, such as a tree of 4 levels and 2 branches or a cube of dimension 4, the number of SLAM nodes (QUEUEs, ASSIGN statements, GOON statements and CREATE statements) in the program exceeded the limit of 500 in the SLAM simulator. This entailed making changes in the SLAM source program to increase the dimension of certain arrays and recompiling it.

# CHAPTER 4

# DISCUSSION OF THE RESULTS

## 4-1 Comparison of Analytical Results with Simulation

We are now in a position to compare the analytical results of Chapter 3 with the simulation results (case I) of Chapter 4. Again, $\lambda$ = 1 message/unit time and $\mu$ = 20 messages/unit time.

The percentage error for the analytical results is calculated as follows:

$$\% \text{ error} = \frac{\text{simulation result} - \text{analytical result}}{\text{simulation result}} \times 100$$

Tables 4.1, 4.2 and 4.3 summarize the comparison for the ring, cube and tree networks respectively. The number of significant digits obtained in the simulation results for the cube network does not provide sufficient accuracy for proper comparison, because of round-off error. This is due to the choice of $\lambda$ and $\mu$. Table 4.4 shows a second comparison for the cube network with $\lambda$ =1 message/unit time and $\mu$ = 8 messages/unit time.

It may be noted that the percentage error is less than

Table 4.1

Comparison: Ring Network

| No. of Nodes N | Mean Queue Length (no. of messages) | | | Mean Time in System (time units) | | |
|---|---|---|---|---|---|---|
| | By Simulation | By Analysis | % Error | By Simulation | By Analysis | % Error |
| 5 | 0.00144 | 0.00146 | -1.39 | 0.07597 | 0.07793 | -2.58 |
| 6 | 0.00208 | 0.00212 | -1.92 | 0.09203 | 0.09425 | -2.41 |
| 8 | 0.00344 | 0.00392 | -13.95 | 0.1190 | 0.1217 | -2.27 |
| 16 | 0.0133 | 0.0127 | 4.51 | 0.2343 | 0.2388 | -1.92 |
| 20 | 0.0215 | 0.0199 | 7.44 | 0.2992 | 0.3030 | -1.27 |

Table 4.2

Comparison: Cube Network

| No. of Nodes N | Mean Queue Length (no. of messages) | | | Mean Time in System (time units) | | |
|---|---|---|---|---|---|---|
| | By Simulation | By Analysis | % Error | By Simulation | By Analysis | % Error |
| 8 | 0.000858 | 0.000840 | 2.09 | 0.08677 | 0.08823 | -1.68 |
| 16 | 0.000790 | 0.000731 | 7.47 | 0.1067 | 0.1096 | -2.72 |

Table 4.3

Comparison: Tree Network

| No. of Levels L | No. of Branches K | Bus Level | Mean Queue Length (no. of messages) | | | Mean Time in System (time units) | | |
|---|---|---|---|---|---|---|---|---|
| | | | By Simulation | By Analysis | % Error | By Simulation | By Analysis | % Error |
| 2 | 2 | 1 | 0.0319 | 0.0330 | -3.45 | 0.1349 | 0.1384 | -2.59 |
| | | 2 | 0.0198 | 0.0200 | -1.01 | | | |
| 2 | 3 | 1 | 0.0939 | 0.0930 | 0.96 | 0.1726 | 0.1752 | -1.51 |
| | | 2 | 0.1753 | 0.1710 | 2.45 | | | |
| 3 | 2 | 1 | 0.0416 | 0.0420 | -0.96 | 0.2463 | 0.2511 | -1.95 |
| | | 2 | 0.1156 | 0.1140 | 1.38 | | | |
| | | 3 | 0.0658 | 0.0670 | -1.82 | | | |
| 2 | 4 | 1 | 0.2113 | 0.2020 | 4.40 | 0.2616 | 0.2517 | 3.78 |
| | | 2 | 1.3218 | 1.1370 | 13.98 | | | |
| 4 | 2 | 1 | 0.0481 | 0.0460 | 4.37 | 0.4366 | 0.4419 | -1.21 |
| | | 2 | 0.1870 | 0.1830 | 2.14 | | | |
| | | 3 | 0.6298 | 0.6090 | 3.30 | | | |
| | | 4 | 0.3220 | 0.3170 | 1.55 | | | |

Table 4.4

Comparison: Cube Network with Slower Service Rate

| No. of Nodes N | Mean Queue Length (no. of messages) | | | Mean Time in System (time units) | | |
|---|---|---|---|---|---|---|
| | By Simulation | By Analysis | % Error | By Simulation | By Analysis | % Error |
| 8 | 0.005613 | 0.005494 | 2.12 | 0.2293 | 0.2307 | -0.61 |
| 16 | 0.004905 | 0.004761 | 2.94 | 0.2831 | 0.2856 | -0.88 |

10 % in most cases. This proves that the analytical results are fairly accurate. It is well to remember that the simulations themselves only approximate the real systems and the assumption of steady-state in a finite time adds some inaccuracy.

## 4-2 Comparison of the Three Interconnection Schemes

One of the reasons that motivated this work was to be able to compare the performance of various interconnection schemes. From the tables in the previous section which summarize the results, the following general observations may be made.

In general, the mean queue length is the largest for the tree configuration. This result should be obvious from the fact that a given bus in a tree carries a much larger message traffic than any node in the other two configurations. The ring network has the next highest mean queue length. Intuitively, the reason for this is that a random message travels through N/4 nodes on the average, contributing to all these queues, whereas in the cube network, the average distance, d/2, is much smaller.

The mean time a message spends in the system is of the same order of magnitude for all configurations

(approximately), although it is higher again for the tree network. This is at least partly due to the longer waiting times at each bus (queue). Similarly, for equal number of nodes, the ring network has a longer TIS than the cube for the same reason. The other factor contributing to TIS is the average distance traveled. This is definitely longer for the ring in comparison to the cube. For the tree, even if the average distance might be smaller in some cases, the longer waiting times result in a longer time in the system.

# CHAPTER 5

# CONCLUSION

This thesis has presented analytical models for three types of microcomputer networks. The results obtained from these models have been verified through simulation for networks with up to 20 nodes. Thus, we have a set of analytical results which could presumably be used to estimate the performance of fairly large networks without the need for expensive simulations.

These results could be valuable tools in the design of large multi-microcomputer systems. The mean queue length provides an indication of the size of the buffer required to store the waiting messages. Since the maximum queue length would exceed the mean, messages that can not be received by a fully occupied buffer would be retransmitted. It would be best to increase the mean queue length by a safety factor for the buffer design, so that performance degradation is minimal.

The mean time spent in the system by a message is an indication of how good the system looks to the user. It is purely a performance measure.

There are some topics related to this thesis that have potential for future research. One such subject is studying the performance with the assumption of finite buffer

capacities. Then the capacities could be varied until an optimum value is obtained which gives the maximum performance to cost ratio. Another extension of this work would be to study a hierarchical system with computing nodes replacing the buses in a tree. In this configuration, computers at the base are low-end processors with limited capabilities while those at higher levels in the hierarchy are more powerful so as to be able to handle the higher message traffic, decision- making and computational needs.

## APPENDIX A

## SAMPLE PROGRAM FOR A RING NETWORK

This Appendix contains a sample program written in SLAM for the simulation of a ring network of 6 nodes as shown in Fig. 1.3. The anticlockwise direction has been chosen for simulation. Detailed comments are provided for node 1. All other nodes are similar.

```
GEN,KUMAR VENKAT,RING OF 6 NODES,4/8/84,1;
LIMITS,6,2,500; 6 QUEUES, 2 ATTRIBUTES PER ENTITY,
 500 CONCURRENT ENTITIES MAXIMUM
NETWORK;
 NODE 1 CREATES MESSAGES TO BE SENT TO NODES
 2, 3 AND 4. DESTINATION NUMBER STORED IN
 ATRIBUTE(2) OF EACH MESSAGE. TOTAL
 MESSAGE GEN RATE IN ONE DIRECTION IS
 0.5 MESSAGE/UNIT TIME
   CREATE,EXPON(5.),0,1,,1;  NODE 1
   ASSIGN,ATRIB(2)=2;DESTINATION = 2
   ACT,,,N1;
   CREATE,EXPON(5.),0,1,,1;
   ASSIGN,ATRIB(2)=3;
   ACT,,,N1;
   CREATE,EXPON(10.),0,1,,1;
   ASSIGN,ATRIB(2)=4;
   ACT,,,N1;
N1  QUEUE(1); QUEUE AT NODE 1
   ACT(1)/1,EXPON(0.05); SERVICE ACTIVITY
   GOON;
   ACT,,ATRIB(2).EQ.2,OUT; ROUTING DECISIONS
   ACT,,ATRIB(2).NE.2,N2; TERMINATE
    MESSAGE IF DESTINATION IS NEXT NODE
    ELSE ADD MESSAGE TO NEXT QUEUE
   CREATE,EXPON(5.),0,1,,1;NODE 2
   ASSIGN,ATRIB(2)=3;
   ACT,,,N2;
   CREATE,EXPON(5.),0,1,,1;
   ASSIGN,ATRIB(2)=4;
   ACT,,,N2;
   CREATE,EXPON(10.),0,1,,1;
```

```
        ASSIGN,ATRIB(2)=5;
        ACT,,,N2;
N2       QUEUE(2);
        ACT(1)/2,EXPON(0.05);
        GOON;
        ACT,,ATRIB(2).EQ.3,OUT;
        ACT,,ATRIB(2).NE.3,N3;
        CREATE,EXPON(5.),0,1,,1;NODE 3
        ASSIGN,ATRIB(2)=4;
        ACT,,,N3;
        CREATE,EXPON(5.),0,1,,1;
        ASSIGN,ATRIB(2)=5;
        ACT,,,N3;
        CREATE,EXPON(10.),0,1,,1;
        ASSIGN,ATRIB(2)=6;
        ACT,,,N3;
N3       QUEUE(3);
        ACT(1)/3,EXPON(0.05);
        GOON;
        ACT,,ATRIB(2).EQ.4,OUT;
        ACT,,ATRIB(2).NE.4,N4;
        CREATE,EXPON(5.),0,1,,1;NODE 4
        ASSIGN,ATRIB(2)=5;
        ACT,,,N4;
        CREATE,EXPON(5.),0,1,,1;
        ASSIGN,ATRIB(2)=6;
        ACT,,,N4;
        CREATE,EXPON(10.),0,1,,1;
        ASSIGN,ATRIB(2)=1;
        ACT,,,N4;
N4       QUEUE(4);
        ACT(1)/4,EXPON(0.05);
        GOON;
        ACT,,ATRIB(2).EQ.5,OUT;
        ACT,,ATRIB(2).NE.5,N5;
        CREATE,EXPON(5.),0,1,,1;   NODE 5
        ASSIGN,ATRIB(2)=6;
        ACT,,,N5;
        CREATE,EXPON(5.),0,1,,1;
        ASSIGN,ATRIB(2)=1;
        ACT,,,N5;
        CREATE,EXPON(10.),0,1,,1;
        ASSIGN,ATRIB(2)=2;
        ACT,,,N5;
N5       QUEUE(5);
        ACT(1)/5,EXPON(0.05);
        GOON;
        ACT,,ATRIB(2).EQ.6,OUT;
        ACT,,ATRIB(2).NE.6,N6;
```

```
        CREATE,EXPON(5.),0,1,,1;  NODE 6
        ASSIGN,ATRIB(2)=1;
        ACT,,,N6;
        CREATE,EXPON(5.),0,1,,1;
        ASSIGN,ATRIB(2)=2;
        ACT,,,N6;
        CREATE,EXPON(10.),0,1,,1;
        ASSIGN,ATRIB(2)=3;
        ACT,,,N6;
N6       QUEUE(6);
        ACT(1)/6,EXPON(0.05);
        GOON;
        ACT,,ATRIB(2).EQ.1,OUT;
        ACT,,ATRIB(2).NE.1,N1;
OUT      COLCT,INT(1),TIME IN SYSTEM; COLECT STATISTICS
        TERM; TERMINATE MESSAGES
        END;
INIT,0,10000; SIMULATION FOR 10000 TIME UNITS
FIN;
```

## APPENDIX B

## SAMPLE PROGRAM FOR A CUBE NETWORK

This Appendix contains a sample program written in SLAM for the simulation of a cube network with d = 3. Fig. 1.4 shows such a network. Comments are provided for the first system node. All other nodes are similar.

```
GEN,KUMAR VENKAT,CUBE NETWORK 3D,3/25/84,1;
LIMITS,24,2,500;24 QUEUES,2 ATTRIBUTES PER ENTITY
;500 CONCURRENT ENTITIES MAXIMUM
 NETWORK;
    NODE 1 CREATES MESSAGES TO BE SENT TO
    NODES 2 THROUGH 8 AND ATTACHES DEST
    ADDRESSES AS ATTRIBUTES.TOTAL MESSAGE
    GEN RATE IS 1 MESSAGE/UNIT TIME
   CREATE,EXPON(7.),0,1,,1; NODE 1
   ASSIGN,ATRIB(2)=2; DESTINATION = 2
   ACT,,,N12;
   CREATE,EXPON(7.),0,1,,1;
   ASSIGN,ATRIB(2)=3;
   ACT,,0.5,N12;
   ACT,,0.5,N13;
   CREATE,EXPON(7.),0,1,,1;
   ASSIGN,ATRIB(2)=4;
   ACT,,,N13;
   CREATE,EXPON(7.),0,1,,1;
   ASSIGN,ATRIB(2)=5;
   ACT,,,N11;
   CREATE,EXPON(7.),0,1,,1;
   ASSIGN,ATRIB(2)=6;
   ACT,,0.5,N12;
   ACT,,0.5,N11;
   CREATE,EXPON(7.),0,1,,1;
   ASSIGN,ATRIB(2)=7;
   ACT,,0.333,N11;
   ACT,,0.333,N12;
   ACT,,0.334,N13;
   CREATE,EXPON(7.),0,1,,1;
   ASSIGN,ATRIB(2)=8;
   ACT,,0.5,N11;
   ACT,,0.5,N13;
```

```
N11     QUEUE(1);QUEUE 1 OF NODE 1
        ACT(1)/1,EXPON(0.05);
        GOON;
        ACT,,ATRIB(2).EQ.5,OUT; ROUTING DECISIONS
        ACT,,ATRIB(2).EQ.8,N51;
        ACT,,ATRIB(2).EQ.6,N53;
        ACT,,ATRIB(2).EQ.7,T11;
N12     QUEUE(2); QUEUE 2 OF NODE 1
        ACT(1)/2,EXPON(0.05);
        GOON;
        ACT,,ATRIB(2).EQ.2,OUT;
        ACT,,ATRIB(2).EQ.3,N22;
        ACT,,ATRIB(2).EQ.6,N23;
        ACT,,ATRIB(2).EQ.7,T12;
N13     QUEUE(3); QUEUE 3 OF NODE 1
        ACT(1)/3,EXPON(0.05);
        GOON;
        ACT,,ATRIB(2).EQ.4,OUT;
        ACT,,ATRIB(2).EQ.3,N43;
        ACT,,ATRIB(2).EQ.8,N41;
        ACT,,ATRIB(2).EQ.7,T13;
T11     GOON; FOR EQUALLY SHORT PATHS,
        ACT,,0.5,N51; ONE CHOSEN WITH
        ACT,,0.5,N53;  UNIFORM PROBABILITY
T12     GOON;
        ACT,,0.5,N22;
        ACT,,0.5,N23;
T13     GOON;
        ACT,,0.5,N41;
        ACT,,0.5,N43;
        CREATE,EXPON(7.),0,1,,1; NODE 2
        ASSIGN,ATRIB(2)=1;
        ACT,,,N21;
        CREATE,EXPON(7.),0,1,,1;
        ASSIGN,ATRIB(2)=3;
        ACT,,,N22;
        CREATE,EXPON(7.),0,1,,1;
        ASSIGN,ATRIB(2)=4;
        ACT,,0.5,N22;
        ACT,,0.5,N21;
        CREATE,EXPON(7.),0,1,,1;
        ASSIGN,ATRIB(2)=5;
        ACT,,0.5,N21;
        ACT,,0.5,N23;
        CREATE,EXPON(7.),0,1,,1;
        ASSIGN,ATRIB(2)=6;
        ACT,,,N23;
        CREATE,EXPON(7.),0,1,,1;
        ASSIGN,ATRIB(2)=7;
```

```
        ACT,,0.5,N22;
        ACT,,0.5,N23;
        CREATE,EXPON(7.),0,1,,1;
        ASSIGN,ATRIB(2)=8;
        ACT,,0.333,N21;
        ACT,,0.333,N22;
        ACT,,0.334,N23;
N21      QUEUE(4);
        ACT(1)/4,EXPON(0.05);
        GOON;
        ACT,,ATRIB(2).EQ.1,OUT;
        ACT,,ATRIB(2).EQ.4,N13;
        ACT,,ATRIB(2).EQ.5,N11;
        ACT,,ATRIB(2).EQ.8,T21;
N22      QUEUE(5);
        ACT(1)/5,EXPON(0.05);
        GOON;
        ACT,,ATRIB(2).EQ.3,OUT;
        ACT,,ATRIB(2).EQ.4,N31;
        ACT,,ATRIB(2).EQ.7,N33;
        ACT,,ATRIB(2).EQ.8,T22;
N23      QUEUE(6);
        ACT(1)/6,EXPON(0.05);
        GOON;
        ACT,,ATRIB(2).EQ.6,OUT;
        ACT,,ATRIB(2).EQ.7,N63;
        ACT,,ATRIB(2).EQ.5,N61;
        ACT,,ATRIB(2).EQ.8,T23;
T21      GOON;
        ACT,,0.5,N11;
        ACT,,0.5,N13;
T22      GOON;
        ACT,,0.5,N31;
        ACT,,0.5,N33;
T23      GOON;
        ACT,,0.5,N61;
        ACT,,0.5,N63;
        CREATE,EXPON(7.),0,1,,1;      NODE 3
        ASSIGN,ATRIB(2)=1;
        ACT,,0.5,N31;
        ACT,,0.5,N32;
        CREATE,EXPON(7.),0,1,,1;
        ASSIGN,ATRIB(2)=2;
        ACT,,,N32;
        CREATE,EXPON(7.),0,1,,1;
        ASSIGN,ATRIB(2)=4;
        ACT,,,N31;
        CREATE,EXPON(7.),0,1,,1;
        ASSIGN,ATRIB(2)=5;
```

```
        ACT,,0.333,N31;
        ACT,,0.333,N32;
        ACT,,0.334,N33;
        CREATE,EXPON(7.),0,1,,1;
        ASSIGN,ATRIB(2)=6;
        ACT,,0.5,N32;
        ACT,,0.5,N33;
        CREATE,EXPON(7.),0,1,,1;
        ASSIGN,ATRIB(2)=7;
        ACT,,,N33;
        CREATE,EXPON(7.),0,1,,1;
        ASSIGN,ATRIB(2)=8;
        ACT,,0.5,N31;
        ACT,,0.5,N33;
N31      QUEUE(7);
        ACT(1)/7,EXPON(0.05);
        GOON;
        ACT,,ATRIB(2).EQ.4,OUT;
        ACT,,ATRIB(2).EQ.1,N42;
        ACT,,ATRIB(2).EQ.8,N41;
        ACT,,ATRIB(2).EQ.5,T31;
N32      QUEUE(8);
        ACT(1)/8,EXPON(0.05);
        GOON;
        ACT,,ATRIB(2).EQ.2,OUT;
        ACT,,ATRIB(2).EQ.6,N23;
        ACT,,ATRIB(2).EQ.1,N21;
        ACT,,ATRIB(2).EQ.5,T32;
N33      QUEUE(9);
        ACT(1)/9,EXPON(0.05);
        GOON;
        ACT,,ATRIB(2).EQ.7,OUT;
        ACT,,ATRIB(2).EQ.6,N72;
        ACT,,ATRIB(2).EQ.8,N73
        ACT,,ATRIB(2).EQ.5,T33;
T31      GOON;
        ACT,,0.5,N41;
        ACT,,0.5,N42;
T32      GOON;
        ACT,,0.5,N21;
        ACT,,0.5,N23;
T33      GOON;
        ACT,,0.5,N71;
        ACT,,0.5,N72;
        CREATE,EXPON(7.),0,1,,1;    NODE 4
        ASSIGN,ATRIB(2)=1;
        ACT,,,N42;
        CREATE,EXPON(7.),0,1,,1;
        ASSIGN,ATRIB(2)=2;
```

```
        ACT,,0.5,N43;
        ACT,,0.5,N42;
        CREATE,EXPON(7.),0,1,,1;
        ASSIGN,ATRIB(2)=3;
        ACT,,,N43;
        CREATE,EXPON(7.),0,1,,1;
        ASSIGN,ATRIB(2)=5;
        ACT,,0.5,N41;
        ACT,,0.5,N42;
        CREATE,EXPON(7.),0,1,,1;
        ASSIGN,ATRIB(2)=6;
        ACT,,0.333,N41;
        ACT,,0.333,N42;
        ACT,,0.334,N43;
        CREATE,EXPON(7.),0,1,,1;
        ASSIGN,ATRIB(2)=7;
        ACT,,0.5,N41;
        ACT,,0.5,N43;
        CREATE,EXPON(7.),0,1,,1;
        ASSIGN,ATRIB(2)=8;
        ACT,,,N41;
N41      QUEUE(10);
        ACT(1)/10,EXPON(0.05);
        GOON;
        ACT,,ATRIB(2).EQ.8,OUT;
        ACT,,ATRIB(2).EQ.5,N81;
        ACT,,ATRIB(2).EQ.7,N83;
        ACT,,ATRIB(2).EQ.6,T41;
N42      QUEUE(11);
        ACT(1)/11,EXPON(0.05);
        GOON;
        ACT,,ATRIB(2).EQ.1,OUT;
        ACT,,ATRIB(2).EQ.2,N12;
        ACT,,ATRIB(2).EQ.5,N11;
        ACT,,ATRIB(2).EQ.6,T42;
N43      QUEUE(12);
        ACT(1)/12,EXPON(0.05);
        GOON;
        ACT,,ATRIB(2).EQ.3,OUT;
        ACT,,ATRIB(2).EQ.2,N32;
        ACT,,ATRIB(2).EQ.7,N33;
        ACT,,ATRIB(2).EQ.6,T43;
T41      GOON;
        ACT,,0.5,N81;
        ACT,,0.5,N83;
T42      GOON;
        ACT,,0.5,N11;
        ACT,,0.5,N12;
T43      GOON;
```

```
        ACT,,0.5,N32;
        ACT,,0.5,N33;
        CREATE,EXPON(7.),0,1,,1;NODE 5
        ASSIGN,ATRIB(2)=1;
        ACT,,,N52;
        CREATE,EXPON(7.),0,1,,1;
        ASSIGN,ATRIB(2)=2;
        ACT,,0.5,N52;
        ACT,,0.5,N53;
        CREATE,EXPON(7.),0,1,,1;
        ASSIGN,ATRIB(2)=3;
        ACT,,0.333,N51;
        ACT,,0.333,N52;
        ACT,,0.334,N53;
        CREATE,EXPON(7.),0,1,,1;
        ASSIGN,ATRIB(2)=4;
        ACT,,0.5,N51;
        ACT,,0.5,N52;
        CREATE,EXPON(7.),0,1,,1;
        ASSIGN,ATRIB(2)=6;
        ACT,,,N53;
        CREATE,EXPON(7.),0,1,,1;
        ASSIGN,ATRIB(2)=7;
        ACT,,0.5,N51;
        ACT,,0.5,N53;
        CREATE,EXPON(7.),0,1,,1;
        ASSIGN,ATRIB(2)=8;
        ACT,,,N51;
N51      QUEUE(13);
        ACT(1)/13,EXPON(0.05);
        GOON;
        ACT,,ATRIB(2).EQ.8,OUT;
        ACT,,ATRIB(2).EQ.4,N82;
        ACT,,ATRIB(2).EQ.7,N83;
        ACT,,ATRIB(2).EQ.3,T51;
N52      QUEUE(14);
        ACT(1)/14,EXPON(0.05);
        GOON;
        ACT,,ATRIB(2).EQ.1,OUT;
        ACT,,ATRIB(2).EQ.2,N12;
        ACT,,ATRIB(2).EQ.4,N13;
        ACT,,ATRIB(2).EQ.3,T52;
N53      QUEUE(15);
        ACT(1)/15,EXPON(0.05);
        GOON;
        ACT,,ATRIB(2).EQ.6,OUT;
        ACT,,ATRIB(2).EQ.2,N62;
        ACT,,ATRIB(2).EQ.7,N63;
        ACT,,ATRIB(2).EQ.3,T53;
```

```
T51     GOON;
        ACT,,0.5,N82;
        ACT,,0.5,N83;
T52     GOON;
        ACT,,0.5,N12;
        ACT,,0.5,N13;
T53     GOON;
        ACT,,0.5,N63;
        ACT,,0.5,N62;
        CREATE,EXPON(7.),0,1,,1;   NODE 6
        ASSIGN,ATRIB(2)=1;
        ACT,,0.5,N61;
        ACT,,0.5,N62;
        CREATE,EXPON(7.),0,1,,1;
        ASSIGN,ATRIB(2)=2;
        ACT,,,N62;
        CREATE,EXPON(7.),0,1,,1;
        ASSIGN,ATRIB(2)=3;
        ACT,,0.5,N62;
        ACT,,0.5,N63;
        CREATE,EXPON(7.),0,1,,1;
        ASSIGN,ATRIB(2)=4;
        ACT,,0.333,N61;
        ACT,,0.333,N62;
        ACT,,0.334,N63;
        CREATE,EXPON(7.),0,1,,1;
        ASSIGN,ATRIB(2)=5;
        ACT,,,N61;
        CREATE,EXPON(7.),0,1,,1;
        ASSIGN,ATRIB(2)=7;
        ACT,,,N63;
        CREATE,EXPON(7.),0,1,,1;
        ASSIGN,ATRIB(2)=8;
        ACT,,0.5,N61;
        ACT,,0.5,N63;
N61     QUEUE(16);
        ACT(1)/16,EXPON(0.05);
        GOON;
        ACT,,ATRIB(2).EQ.5,OUT;
        ACT,,ATRIB(2).EQ.1,N52;
        ACT,,ATRIB(2).EQ.8,N51;
        ACT,,ATRIB(2).EQ.4,T61;
N62     QUEUE(17);
        ACT(1)/17,EXPON(0.05);
        GOON;
        ACT,,ATRIB(2).EQ.2,OUT;
        ACT,,ATRIB(2).EQ.1,N21;
        ACT,,ATRIB(2).EQ.3,N22;
        ACT,,ATRIB(2).EQ.4,T62;
```

```
N63     QUEUE(18);
        ACT(1)/18,EXPON(0.05);
        GOON;
        ACT,,ATRIB(2).EQ.7,OUT;
        ACT,,ATRIB(2).EQ.8,N71;
        ACT,,ATRIB(2).EQ.3,N73;
        ACT,,ATRIB(2).EQ.4,T63;
T61     GOON;
        ACT,,0.5,N51;
        ACT,,0.5,N52;
T62     GOON;
        ACT,,0.5,N21;
        ACT,,0.5,N22;
T63     GOON;
        ACT,,0.5,N71;
        ACT,,0.5,N73;
        CREATE,EXPON(7.),0,1,,1;  NODE 7
        ASSIGN,ATRIB(2)=1;
        ACT,,0.333,N71;
        ACT,,0.333,N72;
        ACT,,0.334,N73;
        CREATE,EXPON(7.),0,1,,1;
        ASSIGN,ATRIB(2)=2;
        ACT,,0.5,N72;
        ACT,,0.5,N73;
        CREATE,EXPON(7.),0,1,,1;
        ASSIGN,ATRIB(2)=3;
        ACT,,,N73;
        CREATE,EXPON(7.),0,1,,1;
        ASSIGN,ATRIB(2)=4;
        ACT,,0.5,N73;
        ACT,,0.5,N71;
        CREATE,EXPON(7.),0,1,,1;
        ASSIGN,ATRIB(2)=5;
        ACT,,0.5,N72;
        ACT,,0.5,N71;
        CREATE,EXPON(0.05),0,1,,1;
        ASSIGN,ATRIB(2)=6;
        ACT,,,N72;
        CREATE,EXPON(7.),0,1,,1;
        ASSIGN,ATRIB(2)=8;
        ACT,,,N71;
N71     QUEUE(19);
        ACT(1)/19,EXPON(0.05);
        GOON;
        ACT,,ATRIB(2).EQ.8,OUT;
        ACT,,ATRIB(2).EQ.4,N82;
        ACT,,ATRIB(2).EQ.5,N81;
        ACT,,ATRIB(2).EQ.1,T71;
```

```
N72     QUEUE(20);
        ACT(1)/20,EXPON(0.05);
        GOON;
        ACT,,ATRIB(2).EQ.6,OUT;
        ACT,,ATRIB(2).EQ.5,N61;
        ACT,,ATRIB(2).EQ.2,N62;
        ACT,,ATRIB(2).EQ.1,T72;
N73     QUEUE(21);
        ACT(1)/21,EXPON(0.05);
        GOON;
        ACT,,ATRIB(2).EQ.3,OUT;
        ACT,,ATRIB(2).EQ.4,N31;
        ACT,,ATRIB(2).EQ.2,N32;
        ACT,,ATRIB(2).EQ.1,T73;
T71     GOON;
        ACT,,0.5,N31;
        ACT,,0.5,N82;
T72     GOON;
        ACT,,0.5,N61;
        ACT,,0.5,N62;
T73     GOON;
        ACT,,0.5,N31;
        ACT,,0.5,N32;
        CREATE,EXPON(7.),0,1,,1;NODE 8
        ASSIGN,ATRIB(2)=1;
        ACT,,0.5,N81;
        ACT,,0.5,N82;
        CREATE,EXPON(7.),0,1,,1;
        ASSIGN,ATRIB(2)=2;
        ACT,,0.333,N81;
        ACT,,0.333,N82;
        ACT,,0.334,N83;
        CREATE,EXPON(7.),0,1,,1;
        ASSIGN,ATRIB(2)=3;
        ACT,,0.5,N82;
        ACT,,0.5,N83;
        CREATE,EXPON(7.),0,1,,1;
        ASSIGN,ATRIB(2)=4;
        ACT,,,N81;
        CREATE,EXPON(7.),0,1,,1;
        ASSIGN,ATRIB(2)=5;
        ACT,,,N81;
        CREATE,EXPON(7.),0,1,,1;
        ASSIGN,ATRIB(2)=6;
        ACT,,0.5,N81;
        ACT,,0.5,N83;
        CREATE,EXPON(7.),0,1,,1;
        ASSIGN,ATRIB(2)=7;
        ACT,,,N83;
```

```
N81     QUEUE(22);
        ACT(1)/22,EXPON(0.05);
        GOON;
        ACT,,ATRIB(2).EQ.5,OUT;
        ACT,,ATRIB(2).EQ.1,N52;
        ACT,,ATRIB(2).EQ.6,N53;
        ACT,,ATRIB(2).EQ.2,T81;
N82     QUEUE(23);
        ACT(1)/23,EXPON(0.05);
        GOON;
        ACT,,ATRIB(2).EQ.4,OUT;
        ACT,,ATRIB(2).EQ.1,N42;
        ACT,,ATRIB(2).EQ.3,N43;
        ACT,,ATRIB(2).EQ.2,T82;
N83     QUEUE(24);
        ACT(1)/24,EXPON(0.05);
        GOON;
        ACT,,ATRIB(2).EQ.7,OUT;
        ACT,,ATRIB(2).EQ.6,N72;
        ACT,,ATRIB(2).EQ.3,N73;
        ACT,,ATRIB(2).EQ.2,T83;
T81     GOON;
        ACT,,0.5,N52;
        ACT,,0.5,N53;
T82     GOON;
        ACT,,0.5,N42;
        ACT,,0.5,N43;
T83     GOON;
        ACT,,0.5,N72;
        ACT,,0.5,N73;
OUT     COLCT,INT(1),TIME IN SYSTEM;COLLECT STATISTICS
        TERM; TERMINATE MESSAGES
        END;
INIT,0,10000; SIMULATION FOR 10000 TIME UNITS
FIN;
```

## APPENDIX C

## SAMPLE PROGRAM FOR A TREE NETWORK

This Appendix contains a sample program written in SLAM for the simulation of a tree network with L=2 and K=2, as shown in Fig. 1.5(a).

```
GEN,KUMAR VENKAT,TREE OF L2 K2,3/17/84,1;
LIMITS,3,2,500; 3 QUEUES, 2 ATTRIBUTES PER ENTITY,
; 500 CONCURRENT ENTITIES MAXIMUM.
NETWORK;
     EACH NODE CREATES MESSAGES FOR ALL
     OTHER NODES AT THE SAME RATE. TOTAL
     RATE IS 1 MESSAGE/TIME UNIT.
   CREATE,EXPON(3.),0,1,,1; NODE 1
   ASSIGN,ATR(2)=2;DESTINATION=2
   ACT,,,N1;
   CREATE,EXPON(3.),0,1,,1;
   ASSIGN,ATR(2)=3;
   ACT,,,N1;
   CREATE,EXPON(3.),0,1,,1;
   ASSIGN,ATR(2)=4;
   ACT,,,N1;
   CREATE,EXPON(3.),0,1,,1; NODE 2
   ASSIGN,ATR(2)=1;
   ACT,,,N1;
   CREATE,EXPON(3.),0,1,,1;
   ASSIGN,ATR(2)=3;
   ACT,,,N1;
   CREATE,EXPON(3.),0,1,,1;
   ASSIGN,ATR(2)=4;
   ACT,,,N1;
   CREATE,EXPON(3.),0,1,,1; NODE 3
   ASSIGN,ATR(2)=1;
   ACT,,,N2;
   CREATE,EXPON(3.),0,1,,1;
   ASSIGN,ATR(2)=2;
   ACT,,,N2;
   CREATE,EXPON(3.),0,1,,1;
   ASSIGN,ATR(2)=4;
   ACT,,,N2;
   CREATE,EXPON(3.),0,1,,1; NODE 4
   ASSIGN,ATR(2)=1;
```

```
        ACT,,,N2;
        CREATE,EXPON(3.),0,1,,1;
        ASSIGN,ATR(2)=2;
        ACT,,,N2;
        CREATE,EXPON(3.),0,1,,1;
        ASSIGN,ATR(2)=3;
        ACT,,,N2;
N1      QUEUE(1); BUS 1,LEVEL 1
        ACT(1)/1,EXPON(0.05); SERVICE ACTIVITY
        GOON;
        ACT,,ATR(2).EQ.1.OR.ATR(2).EQ.2,OUT; ROUTING
        ACT,,ATR(2).EQ.3.OR.ATR(2).EQ.4,N3;
N2      QUEUE(2); BUS 2, LEVEL 1
        ACT(1)/2,EXPON(0.05);
        GOON;
        ACT,,ATR(2).EQ.3.OR.ATR(2).EQ.4,OUT;
        ACT,,ATR(2).EQ.1.OR.ATR(2).EQ.2,N3;
N3      QUEUE(3); BUS AT LEVEL 2
        ACT(1)/3,EXPON(0.05);
        GOON;
        ACT,,ATR(2).EQ.1.OR.ATR(2).EQ.2,N1;
        ACT,,ATR(2).EQ.3.OR.ATR(2).EQ.4,N2;
OUT     COLCT,INT(1),TIME IN SYSTEM; COLLECT STATISTICS
        TERM; TERMINATE MESSAGES
        END;
      INIT,0,10000; SIMULATION FOR 10000 TIME UNITS
      FIN;
```

## REFERENCES


1. B. W. Arden and H. Lee, "A Regular Network for Multicomputer Systems", IEEE Trans. Computers, vol. c-31, pp. 60-69, January 1982.

2. L. N. Bhuyan and D. P. Agrawal, "Generalized Hypercube and Hyperbus Structures for a Computer Network", IEEE Trans. Computers, vol. c-33, pp. 323-333, April 1984.

3. W. W. Chu, G. Bayolle and D. G. Hibbits, "An Analysis of a Tandem Queueing System for Flow Control in Computer Networks", IEEE Trans. Computers, vol. c-30, pp. 318-324, May 1981.

4. R. A. Finkel and M. H. Solomon, "Processor Interconnection Strategies", IEEE Trans. Computers, vol. c-29, pp. 360-371, May 1980.

5. W. C. Giffin, Queueing: Basic Theory and Applications. Columbus, Ohio: Grid Inc., 1978.

6. J. P. Hayes, Computer Architecture and Organization. New York,N.Y.: McGraw-Hill Book Co., 1978.

7. J. R. Jackson, "Networks of Waiting Lines", Oper. Res., vol. 5, pp.518-521, August 1957.

8. L. Kleinrock, Communication Nets. New York,N.Y.: McGraw-Hill Book Co., 1964.

9. S. S. Lam and Y. C. L. Lien, "Congestion Control of Packet Communication Networks by Input Buffer Limits: A

Simulation Study", IEEE Trans. Computers, vol. c-30, pp.733-742, October 1981.

10. A. A. B. Pritsker and C. D. Pegden, Introduction to Simulation and SLAM. New York, N.Y.: John Wiley & Sons, 1979.

11. W. L. Spetz, "Microprocessor Networks", Computer, vol. 10, 1977.

12. H. Sullivan and T. R. Bashkow, "A Large Scale, Homogenous, Fully Distributed Parallel Machine", Proc. 4 th Symp. Comput. Architecture, pp. 105-124, 1977.

13. A. S. Tanenbaum, Computer Networks. Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1981.

14. L. D. Wittie, "Communication Structures for Large Networks of Microcomputers", IEEE Trans. Computers, vol. c-30, pp.264-273, April 1981.

15. S. B. Wu and M. T. Liu, "A Cluster Structure as an Interconnection Network for Large Multimicrocomputer Systems", IEEE Trans. Computers, vol. c-30, pp.254-264, April 1981.

# VITA

Kumar Venkatasubramaniam was born in Bombay, India, on October 12, 1960, the son of Saraswathi Venkatasubramaniam and Venkatalakshmana Venkatasubramaniam. After attending T.V.S.L. Higher Secondary School, Madurai, India, he completed his high school work at Burbank Senior High School, Burbank, California, in 1977. He entered the Regional Engineering College, University of Madras, India, in 1977 and received the Bachelor of Engineering (Honors) degree in 1982. In January, 1983, he entered the Graduate School of the University of Texas at El Paso.

Permanent address: C-2A Ilakuvanar Street
Tirunagar, Madurai 625006
Tamil Nadu
India

This thesis was typed by Kumar Venkatasubramaniam