National Library of Canada
Bibliothèque nationale du Canada

Canadian Theses Service
Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

NL-339 (r. 88/04) c

Canada

National Library of Canada
Bibliothèque nationale du Canada

Canadian Theses Service
Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-315-53285-8

Canada

A SIMULATION APPROACH FOR ESTIMATING THE PERFORMANCE OF A MULTIPROCESSOR DIGITAL SWITCHING SYSTEM

by

GURSHARAN SIDHU

A thesis submitted to
the School of Graduate Studies and Research
of the University of Ottawa
in partial fulfillment of the requirements
of Master of Science in Systems Science

Ottawa, Ontario, 1989

© Gursharan Sidhu, Ottawa, Canada, 1989.

ABSTRACT

Several different structures for the operation of multiprocessor-based digital switching systems for use in telecommunications industry have been proposed. Their relative performance can be evaluated through the development of an appropriate performance model. The development and use of such a model is the subject of the study in this thesis.

A closed queueing network model which represents the structure of a class of multiprocessor digital switching systems is proposed. Attention is focused on call setup which is defined as a sequence of states associated with telephony tasks and subscriber events. A simulation program which implements the model has been developed and is used to obtain values for performance measures such as the processor utilization, the call-setup time, the call-delay time, the call-throughput and the call-attempts/hour.

By assigning appropriate values to parameters in the simulation program, it can be configured to correspond to any specific architecture. The model's validity is established by tailoring it to one particular existing system and comparing the behavior with the available data for the existing system. The principal application of the model within this

study is to evaluate the performance of two specific multi-processor digital switching system architectures.

ACKNOWLEDGEMENTS

I would like to express my deep sense of gratitude to my supervisor, Professor Louis G. Birta, for his advice, encouragement and valuable guidance throughout the course of my study and research. He was kind and patient and spent considerable amount of his precious time not only directing the research but also during the preparation of this thesis document.

I would like to thank the Computer Science Department of the University of Ottawa for providing the necessary facilities throughout the course of my study.

I am also extremely grateful to my wife, Jyot P. Sidhu, for her unwaivering support and constant encouragement without which it would have been impossible to finish this thesis project. I will always appreciate her incredible tolerance and help.

LIST OF ACRONYMS AND ABBREVIATIONS

CP - Control Processor

DL - Destination Line

LI - Line Interface

MD - Message Distributor

MDSS - Multiprocessor Digital Switching System

OL - Origination Line

PCM - Pulse Code Modulation

RT - Routing Table

SN - Switching Network

SPC - Stored Program Control

TDM - Time Division Multiplexing

LIST OF ILLUSTRATIONS

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

A desirable prerequisite to the construction of large complex systems is a comprehensive understanding of both the dynamic behavior and likely performance characteristics of such systems. The importance of such insight is particularly relevant when implementation costs rise to substantial levels. Computer-based communication systems provide an example of systems where such insight is of crucial importance.

1.1 Modeling And Simulation

There are two broad methods for evaluating the performance of a given system: Experimental Analysis, and Modeling and Simulation. Experimental analysis consists of measuring the performance of the actual system and it can be quite laborious and expensive. It is not feasible at all if the system is in concept phase and hence does not yet exist. On the other hand, the modeling and the simulation approach

undertakes to determine and evaluate the dynamic behavior of a system without directly involving the actual system. It can substantially reduce the time factor in the design process and can be very cost-effective. Simulation is amenable to evaluating alternate system structures. Once a performance model is constructed and validated, it can be effectively used to resolve design issues and predict the behavior of the system. These features and advantages have made modeling and simulation an extremely popular and widely used methodology for evaluating the behavior and performance of complex dynamic systems.

1.2 Performance Of Multiprocessor Systems

Performance evaluation of multiprocessor systems has been reported quite extensively in the literature. Most of these studies have used queueing networks for modeling various types of contention for resources in a multiprocessing environment. Often, an analytic solution is formulated for the queueing model [11,12,14,16]. In situations where analytic solutions becomes very complex or are not feasible, authors have used a simulation methodology to derive performance estimates [6,10,19]. The simulation approach is

based on the construction of a suitable model of the system which reflects its relevant dynamic properties.

It is interesting to note that relatively little has been reported in the literature on the performance analysis of actual systems used in real-time control applications. Most authors have considered general purpose multiprocessor system models, as in [1,3,14]. While the results obtained in such general studies have broad application, they often do not adequately treat the issues that are of particular concern in a specific class of systems. Insight into these issues can be obtained only by specialized studies that focus on the area of concern that are distinctive to the problem class.

In the study outlined in this thesis, attention is focused on a specific class of multiprocessor systems used in real-time control, i.e the Multiprocessor Digital Switching Systems (MDSS) which are used in the telephone industry. The system structure and operation of an MDSS is presented from the standpoint of queueing systems. Based on this understanding, a closed queueing network model is developed to estimate the performance of an MDSS. The model is used to study the impact of system architecture on the performance of an MDSS.

1.3 Multiprocessor Digital Switching System (MDSS)

A Multiprocessor Digital Switching System uses Stored Program Control (SPC) techniques to accomplish the complex task of Call Processing which is outlined below. A typical MDSS is composed of four major subsystems/processors interconnected via high speed links. Each of these subsystems has one or more processing units, private memory, data and program that is executed to perform the specific tasks assigned to it. One of the subsystems acts as a master and is responsible for coordinating and controlling the activities of the remaining subsystems. The processors in the various subsystems communicate with each other by exchanging messages on the links.

The complete call processing function of an MDSS can be divided into three distinct phases:

1. **Call Setup:** This is the initial phase which begins with the system (i.e. the MDSS) receiving an 'off-hook' signal indicating a call request from a telephone line subscriber. The system sets up the control structures, allocates the required resources (voice channel etc.) and signals (applies dial-tone) to the sub-

scriber to proceed with the request. The subscriber supplies the telephone number of another subscriber line to which it wishes to connect. The system processes the digits, finds the destination line, sets up the control structures, allocates the resources (voice channel), establishes the connection between the two line and signals (applies ring-tone) to the destination line subscriber for an incoming call. When the destination line subscriber answers the call (goes off-hook), phase 2 begins.

2. **Talking:** During this phase the system simply supervises the call as it passes information in digital format between the two connected lines. It uses Time Division Multiplexing (TDM) for high speed switching of Pulse Code Modulation (PCM) signals. It converts voice frequency signals into PCM format and multiplexes into a high speed digital bit stream for processing and transmission across the switch. When the system receives the 'on-hook' signal from either

line indicating a request for termination of the call, phase 3 begins.

3. **Call Disconnect:** This is the final phase in which the system releases all the resources allocated to the two lines. The lines return to an idle state.

The system load for the MDSS is composed of the tasks to be executed in the three phases mentioned above. However, phase (2) of the call processing function consists almost entirely of routine hardware functions and hence can be decoupled from real-time processing [6]. Similarly, phase (3) consists of a very short sequence of tasks and can also be treated separately. From a real-time point of view, therefore the most significant and time critical tasks are performed in phase (1). Hence, it is quite reasonable to view processing in phase (1) as the overall load for the system. This is, in fact, the approach we take to model the performance of the MDSS.

1.4 Performance of MDSS

The system architecture and the number of active calls in the system are two of the most crucial factors affecting the performance of an MDSS. The system architecture distributes the call processing functions in the system and if it leads to non-uniform utilization of the resources, it can limit the capacity and the throughput of the system. For example, if one of the subsystems is loaded to its full capacity while the remaining subsystems are still available to accept more work, then the capacity of the system is clearly limited by the capacity of the loaded subsystem.

The number of calls in the system has a significant impact on the performance of an MDSS. As the number of calls grows in the system, contention for resources increases resulting in queue build up at each subsystem and the performance of the system degrades. In some cases, this also results in bottlenecks and hence results in limiting the capacity of the switch.

The performance indices for the MDSS considered in this study are:

Call-setup Time: This is defined as the elapsed time between the moment a call request (off-

hook signal) is received and the moment when it enters the 'talking' phase, excluding the time taken by the subscriber to dial in the digits.

Call-delay Time: This is defined as the total time spent by the call while waiting for the resources.

Call-throughput: This is defined as the number of calls setup by the system in unit time.

Call-attempts/Hour: This is defined as the total number of calls connected/disconnected by the system in a period of one hour.

The values for these performance measures have appeared in the literature [5,20] These have been obtained by conducting experiments with actual hardware. The setup involves 20,000 or more lines connected to a source capable of generating a large number of calls such that there is always a call request waiting to be served as long as the system has free resources and capacity to accept more calls [20]. The complexity of such experiments and the costs involved provide the motivation for the development of the performance model which can provide a convenient and low cost alternative for the performance evaluation of such systems.

1.5 Scope of the Thesis Project

Similarities in the structural organization and operating principles can be readily identified from the analysis of various existing multiprocessor digital switching systems. These similarities suggest the feasibility of developing a unified approach for studying the relative performance of such systems.

This approach takes the form of a closed queueing network model that provides a realistic representation for the operational behavior of this class of systems. Calls are represented as jobs in the queueing network. Values for the various performance indices are obtained by simulator studies based on the queueing network model.

The model developed in this study has been formulated from a study of several existing MDSS's. It is primarily intended to be used as a tool during the design phase of an MDSS. It can also be used to study the impact of both the design changes and the addition of new features in the existing MDSS's.

The presentation in this thesis is organized as follows:

Chapter 2 describes the system architecture and basic principles of operation of an MDSS. It provides a brief description of the major components of an MDSS and the tasks performed by each of them. To establish basis for the proposed model, call flow scenarios for setting up and disconnecting a call, from the standpoint of queueing system, are described.

Chapter 3 describes the queueing network model which is proposed. The simulation methodology relating to the performance determination during the call setup process is described in detail. In this context, a call is characterized by telephony tasks and subscriber events. The telephony tasks are distributed over the four subsystems of an MDSS in order to achieve the call processing function. The way in which the distribution takes place, together with the structural considerations, specifies the specific MDSS being simulated. Implicit in the process, is the specification of a Routing Table which captures the distinct states associated with the call setup procedure as it relates to a particular MDSS. The Routing Table provides the major input to the simulation program. A high-level overview of the simulation program and the data structures used is also included.

In chapter 4, the validity of the model is examined. Performance results for the model, as tailored to a particular MDSS, are compared with actual data for that MDSS ob-

tained from the literature. Some analysis of the model results are provided.

Chapter 5 describes the use of the model to evaluate the performance of two particular MDSS's. The results are presented in both tabular and graphical forms. An analysis of results is given.

Chapter 6 provides a brief summary and conclusions of this thesis project. It also includes directions for the future work.

CHAPTER 2

MULTIPROCESSOR DIGITAL SWITCHING SYSTEM (MDSS)

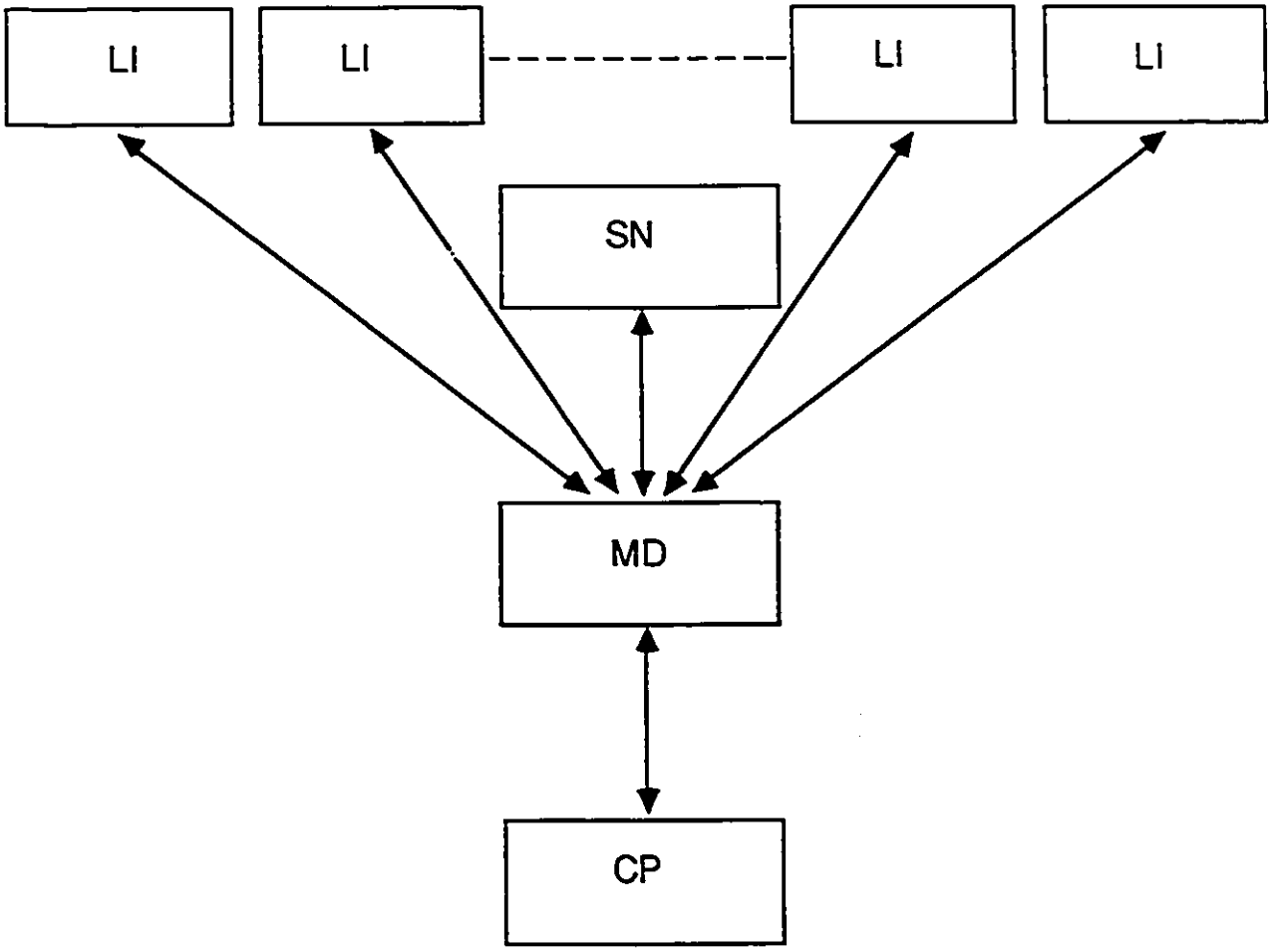
In the last decade, Multiprocessor Digital Switching Systems have evolved very rapidly in the telephone industry. The first MDSS was introduced in 1979 [4]. Today, there are more than 3000 fully digital telephone switching systems operational in North America, Europe and other parts of the world. Most of these MDSS's are Northern Telecom's DMS-100 [4], GTE Automatic Electric Laboratories GTD-5 EAX [5], CIT-ALCATEL's E10.S-TSS.5. [8], and Bell Laboratories No.5 ESS [9]. These are high-capacity switching systems. In comparison with its electro-mechanical predecessor, an MDSS is much more efficient, versatile, reliable and compact. Modern MDSS's have modularity in their hardware and software architecture which facilitates extensions and enhancements. The increasing demand for enhanced telephone service continues to drive the evolution of the digital switching technology.

2.1 System Architecture

Each of the existing MDSS mentioned above uses a distributed processing architecture utilizing microprocessors. It is composed of a number of interacting subsystems each with its own internal architecture. However, the architecture and methods used to perform telephony functions vary from one system to another and are manufacturer dependent.

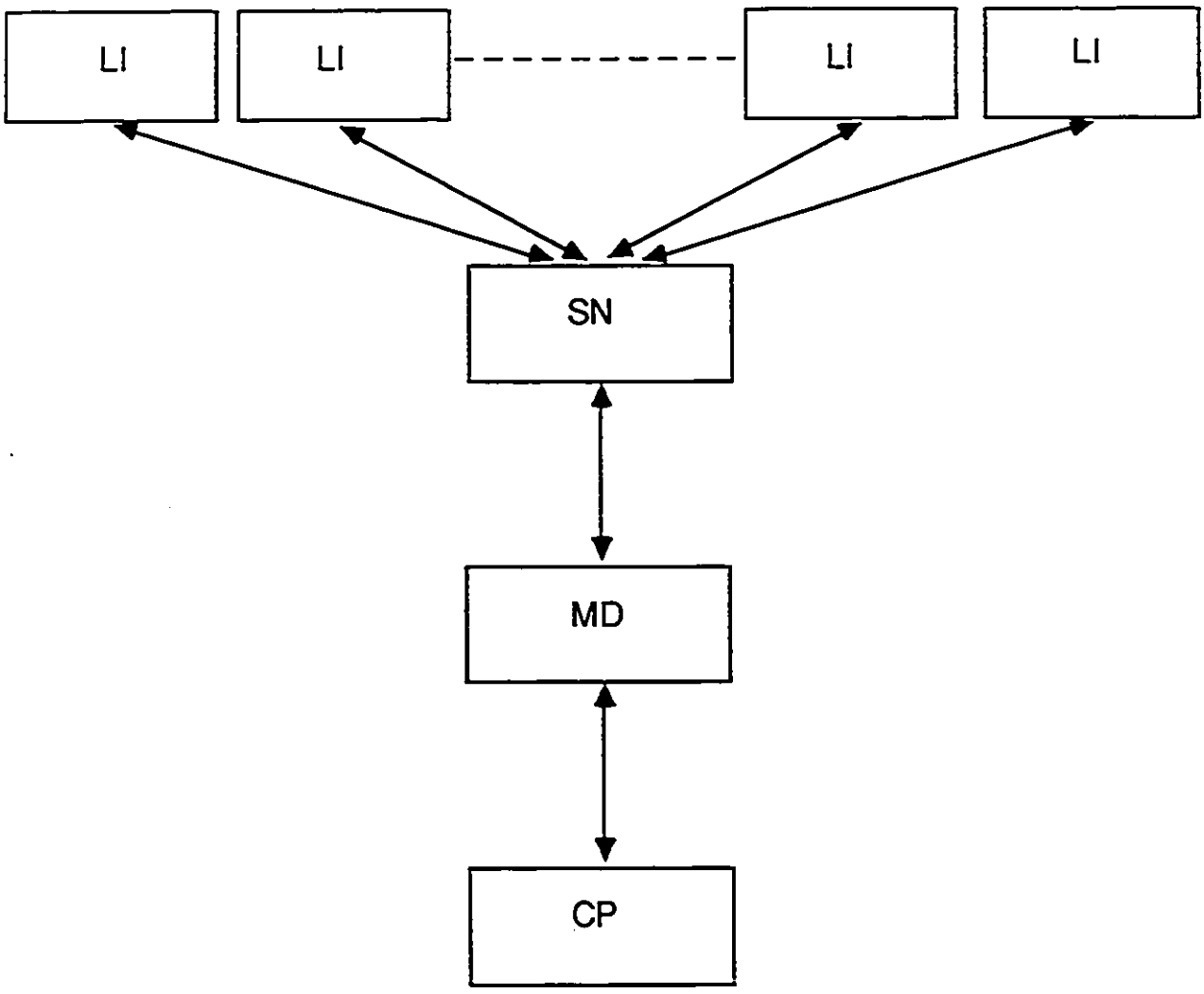
An MDSS within the class which we consider in this study is composed of four major subsystems, which are referred to throughout the sequel as the **Control Processor (CP)**, the **Message Distributor (MD)**, the **Switching Network (SN)** and the **Line Interface (LI)**. Each of the four subsystems consists of a number of identical processors, local memory, data and program store. The subsystems are interconnected using very high speed links. The architecture of such a system depends on the interconnection arrangement used between the subsystems. The two specific system architectures that will be examined extensively in the following chapters are depicted in Figure 1 and Figure 2 and will be referred to as **Architecture-A** and **Architecture-B** respectively.

The architecture of the system distributes the call processing functions (tasks) among the four subsystems. The



CP - Control Processor
MD - Message Distributer
SN - Switching Network
LI - Line Interface
↕ - High Speed Link

Figure 1. Architecture-A



CP - Control Processor
MD - Message Distributer
SN - Switching Network
LI - Line Interface
↕ - High Speed Link

Figure 2. Architecture-B

distribution is based on the hardware/software capabilities provided in a subsystem as well as the type of interface supported by it. A brief description of the subsystems and their functions is given below.

1. Control Processor (CP): The Control Processor is the heart of the MDSS. It coordinates and controls the operation of the remaining subsystems. The CP consists of high speed processing unit(s), memory store and control interface(s). The SN and the LI's communicate with the CP via the Message Distributor. The call processing functions are distributed between the CP, the SN and the LI's. The CP performs the following functions:

   - It sets up the control structures and allocates the resources to the call.
   - It makes use of the configuration and data tables to locate the destination for a given call.
   - It distributes and sequences the call processing functions to be performed by the SN and the LI.
   - It coordinates and controls the activity of the other subsystems and keeps track of their states.

- It is responsible for maintenance and administrative functions for the whole system.

An MDSS is equipped with one or more CP's. The CP's operate either in synchronous mode to provide redundancy or in asynchronous mode to share load. However, in either case, the processing of an individual call (from setup to disconnect) is assigned to a specific CP.

2. Message Distributor (MD): The Message Distributor provides an interface for switching control messages between the CP, the SN and the LI's. The MD consists of one or more processing units and large buffer areas to queue messages. It performs the following functions:

- It shields the CP from the unexpected interrupts caused by the messages coming from the SN and the LI's.
- It distributes the messages originated by the CP for the SN and the LI's. For a given message, it identifies the SN or a particular LI the message is destined. The purpose of this particular function is to hide the physical distribution of the SN and the LI's from the CP.

- It provides fan-out from the CP to the LI's and fan-in from the LI's to the CP.

The functions performed by the MD allow the CP to dedicate its real-time computing capability to tasks such as call processing and resource management. No telephony related processing takes place in the MD.

3. Switching Network (SN): The Switching Network combines both space and time switching techniques to establish interconnections between the origination and destination lines and switches signals between them. Under instructions from the CP the SN performs the following functions:

- It allocates voice channels to the origination and the destination line.
- It establishes interconnections between the voice channels allocated to the origination and the destination line.
- It supervises and maintains the integrity of the established connection for the duration of the call.
- It releases interconnection between channels allocated to the origination and the destination line.

4. Line Interface (LI): The Line Interface provides an interface between the MDSS and the outside world. An MDSS is equipped with different types of LI's to interface with analog lines, analog trunks, digital trunks and possibly Integrated Services Digital Network. The LI performs the following functions:

- It scans lines continuously to detect on/off hook signals from the telephone equipment.
- It reports change in the state of the line to the CP.
- It applies dial/ring tones on the line.
- It collects digits from the line and sends them to the CP.
- It converts analog signals coming from the line into digital format.
- It converts digital signals going out on the line into analog format.

From the analysis of the literature it is apparent that the distribution of the call processing functions is identical for the class of MDSS's that has been examined. In GTD-5 EAX [14], the distribution of the call processing functions between the CP and the LI's is based on transform analysis. In transform analysis terminology, the LI's perform afferent (physical to

logical) and efferent (logical to physical) transform functions. For example, the LI transforms off-hook (electric) signal into a message before sending it to the CP. In the other direction, it transforms a message received from the CP into dial-tone signal on the line. The CP perform logical to logical transform functions.

2.2 Operation

An MDSS is a system which incorporates both distributed processing and centralized control. The basic call processing functions are distributed among various subsystems but the overall control of the system is carried out by the CP. An objective in the distribution of functions is to ensure that the CP is relieved of time consuming tasks. For instance, scanning of the subscriber lines and collection of the digits respectively are continuous and time consuming functions and are assigned to the LI.

The processors in the various subsystems interact with each other by exchanging short messages. A message received by a subsystem contains the information required by that subsystem to identify and execute a particular task. Link

protocols like HDLC, LAPB and some proprietary protocols are implemented to control the flow of messages between processors.

The model development outlined in Chapter 3 is concerned with representing the various events which are involved with the call setup activity within the MDSS. To provide a base for the discussion in Chapter 3, the activities of call setup and call disconnect are described in the following sections.

2.2.1 Call Setup

When a subscriber goes off-hook on one of the lines, the LI detects a change of state on the line. This event marks the origination of a call and is followed by the sequence of events listed below.

- The LI sends an 'off-hook' message to the CP reporting the origination of a call.
- The CP sets up the control structures and allocate the resources to the call. It then sends a message to the SN to assign a voice channel to the origination line.

- The SN assigns a voice channel to the origination line and reports back to the CP.
- The CP updates the state information and sends a message to the LI to apply dial tone and start collecting digits.
- The LI collects the digits dialed by the subscriber and sends them to the CP.
- Using the configuration and data tables, the CP locates the destination line. If the destination line is available, the CP sets up the control structures for the destination line and allocate the resources. The CP then sends a message to the SN to assign a channel to the destination line.
- The SN assigns a voice channel to the destination line and reports back to the CP.
- The CP updates the state information and sends a message to the SN to setup connection between the two voice channels assigned to the origination and the destination line.
- The SN establishes a connection between the two voice channels and reports back to the CP.
- The CP sends one message to the destination LI to apply ring tone on the line and the second to the origination LI to apply an audible ring tone on the line.
- The destination LI applies ring tone and scans for the 'off-hook' state. When the subscriber picks up

the phone, the destination LI detects a change in the state and sends 'off-hook' message to the CP.

- The CP marks the call established. It sends messages to both the origination and the destination LI's to stop the ring tone and look for 'on-hook' state.

At the end of the above sequence of events, the call is setup is in 'talking' state.

2.2.2 Call Disconnect

When either the origination or the destination line hangs up, the respective LI detects a change of state on the line. It marks the event that represents the termination of the call and is followed by the sequence of events listed below.

- The LI sends an 'on-hook' message to the CP.
- The CP sends a message to the SN to take down the connection between the channels assigned to the origination and the destination lines.
- The SN takes down the connections between the two channels and reports back to the CP.

- The CP sends a message to the LI that has reported an 'on-hook' to start scanning for 'off-hook' condition. Similarly the CP sends an equivalent message to the other LI to start scanning for on-hook.
- When the second party hangs up, the associated LI sends 'on-hook' message to the CP.
- The CP releases the resources.

CHAPTER 3

QUEUEING NETWORK MODEL AND SIMULATION

This chapter presents a closed cyclic queueing network model for the class of MDSS's described in the previous chapter. It also describes the simulation approach used to establish values for performance criteria for the queueing network model.

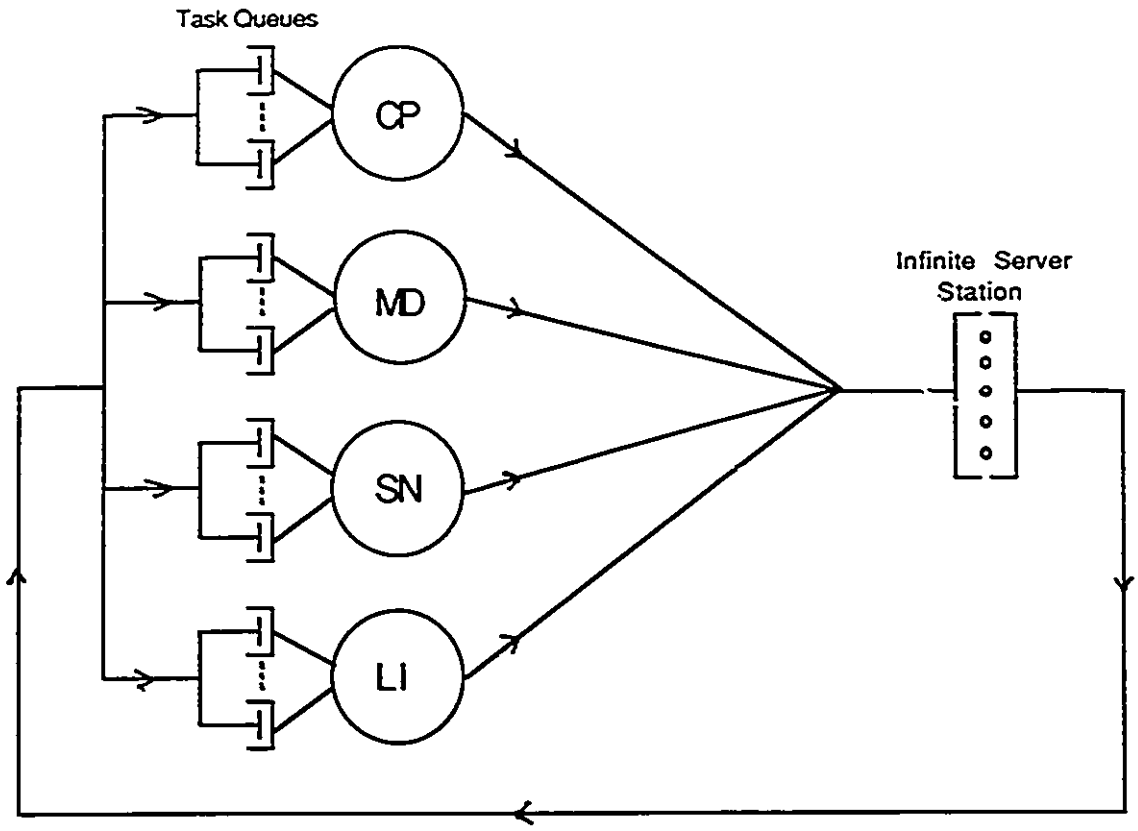
A real-time computer system can be viewed as a combination of two dependent components; namely the controlled process and the controlling computer [17]. Woodbury and Shin [16] have suggested that the development and justification of a performance model for such a system should rely on the workload being modeled as well as the structure of the system handling the workload. Based on this concept, they have proposed a closed queueing network model to study the effects of the workload on the performance for a highly reliable unibus multiprocessor system used in real-time control. The model focuses on a specific structure and takes into account the type of the workload handled by the system.

The development of the queueing network model proposed in this study is based on the same concept. It represents the structure of a class of MDSS's. Call setup is treated as workload for the system. A simulation program simulates the process of call setup in the queueing network and outputs values for several performance measures.

3.1 Queueing Network Model

The performance model proposed is a five-node closed cyclic queueing network, whose structure is shown in Figure 3.

The four subsystems of the MDSS are represented by the four nodes on the left hand side of the figure. To maintain a close relationship between the actual system and the queueing network model, a node is named after the subsystem it represents. Each of the four nodes serves multiple queues, with each queue representing a task assigned to that subsystem. The number of queues served by a subsystem is equal to the number of tasks assigned to that subsystem. This technique facilitates the modelling of all the time consuming functions and the queueing delays on a per task basis. One of the queues for the LI node is a priority



CP - Control Processor
MD - Message Distributer
SN - Switching Network
LI - Line Interface

Figure 3. Queueing Network Model

queue that represents tasks involving the subscriber events. The purpose of the priority queue is to ensure that events from outside the system are handled as quickly as possible. On completion of a task, the LI always scans and processes the priority queue before moving to any other queue.

The queues are served on FCFS basis. All messages in a given queue are assumed to have same distribution for their internal processing time. The processing time distribution for all the queues is taken to be exponentially distributed random variable.

The subscriber events are random in nature and are assumed to have exponentially distributed event times. These are always associated with the tasks to be performed by the LI. To avoid excessive complexity, the subscriber events are included in the processing time for the LI if required.

The node on the right hand side of Figure 3 represents a infinite-server node [14]. It provides a mechanism for modeling the arrival process for messages destined to the four subsystems. Because of the infinite server property, no queueing takes place at this node. The node simply represents inter-processor communication delay between any two subsystems. However, considering the speed of the links connecting the subsystems and the short length of messages, the inter-processor communication delay is negligible and,

hence, can be ignored. There is no concept of separate task queues at this node, all the messages are treated alike.

3.2 Simulation

A simulation approach is used to obtain the desired performance estimates for the queueing network model. The simulation program simulates call setup in the queueing network model presented in the previous section and outputs performance estimates.

The simulation approach is taken because the queueing network does not have a product form solution. The two main characteristics of the queueing network that violate the product form solution are: (1) the priority discipline used for serving queues and (2) the state dependent routing. Furthermore, it is not possible to construct a Markovian model for the performance analysis because the number of states needed is very large.

Computer-based experimentation (i.e. simulation) with the queueing network model provides a very cost-effective means for studying the dynamic behavior and performance of a class of MDSS's. Furthermore, the flexibility inherent in

the associated simulation program facilitates the evaluation of design changes in the existing or proposed MDSS.

3.2.1 Definition of a Call

A call is defined as a sequence of states associated with the telephony tasks and the subscriber events. The telephony tasks are the basic call processing functions to be performed by the system and are characterized by the real-time requirements that represent code segment execution time. The subscriber events consist of initiating a call by going 'off-hook', dialing in telephone number (digits) of the destination line after receiving dial-tone and going 'off-hook' when the line rings. These events are random in duration.

3.2.2 Call Setup

A call is setup in a fixed number of steps that are characterized by telephony tasks and the subscriber events. The number of steps required to setup the call is equal to

the number of distinct telephony tasks to be performed by the four subsystems. The telephony tasks are performed in a specific order which is dependent on the architecture of the system as well as the distribution of the telephony functions in the system. The number and the sequence of steps is represented by a fix sequence of states each associated with a telephony task or the subscriber event. In order to setup a call in a given MDSS, the call must go through the given sequence of states for that MDSS.

Within the class of the MDSS's described in Chapter 2, there is a total of fourteen telephony tasks that are required to be performed in order to setup a call. These fourteen telephony tasks are represented by a fix sequence of fourteen states. Figure 4 presents the relationship between the states and the telephony tasks.

In Figure 4 the number of the telephony tasks is less than the number of the states required to setup a call. This is because some tasks are repeated in more than one states. For example, 'allocate voice channel' is a task performed by the SN twice, once for the origination and once for the destination line.

SN	Task To Be Performed	TN
S1	Detect Off-Hook (OL)	T1
S2	Setup Control Structures, Allocate Resources (OL)	T2
S3	Allocate Voice Channel (OL)	T3
S4	Update Device State (OL)	T4
S5	Apply Dial Tone, Collect Digits (OL)	T5
S6	Process Digits, Locate Destination (OL)	T6
S7	Allocate Voice Channel (DL)	T3
S8	Update Device State (DL)	T4
S9	Setup Connection between Channels (OL,DL)	T7
S10	Update Device State (OL,DL)	T4
S11	Apply Ring Tone, Detect Off-Hook (DL)	T8
S12	Update Device State (OL,DL)	T4
S13	Detect On-Hook (OL,DL)	T9
S14	Talking State	—

SN – State Number

TN – Task Number

OL – Origination Line

DL – Destination Line

Figure 4. Relationship between the States and the Tasks

3.2.3 Distribution of the Telephony Tasks

The architecture of the system distributes the call processing functions (tasks) among the four subsystems. The distribution is based on the hardware/software capabilities provided in a subsystem as well as the type of interface supported by it. From the analysis of the literature it is apparent that the distribution of the telephony tasks associated with the call setup process is identical for the class of MDSS's that has been examined.

The distribution of the telephony tasks for call setup process is described in Figure 5. A particular task is assigned to a subsystem marked with 'X'.

Task Number	Task Assigned To Subsystem			
	CP	MD	SN	LI
T1				X
T2	X			
T3			X	
T4	X			
T5				X
T6	X			
T7			X	
T8				X
T9				X
T10		X		
T11		X		

Figure 5. Distribution of the Telephony Tasks for Architecture-A

The two telephony tasks (T10, T11) listed in Figure 5 that are not included in Figure 4 are the tasks performed by the MD. As described in chapter 2, the MD does not perform any telephony task, it simply acts as a distributing center for the messages. It forwards the messages received from the SN and the LI's to the CP (this is task T10) and distributes

the messages received from the CP to the SN and the LI's (this is task T11). Hence, these two tasks are executed for each message flowing into and out of the CP.

3.2.4 Routing Table (RT)

The Routing Table is the most important input to the simulation program. It describes the distribution of telephony tasks among the four subsystems, the specific order in which the tasks are to be performed and flow of messages within the queueing network model. The RT is dependent on the architecture of an MDSS and it differs for one architecture to another. The following section outlines the construction of the RT for Architecture-A described in Chapter 2.

Based on Figure 4 and Figure 5 and the assumed architecture (i.e. Architecture-A), the complete scenario for setting up a call can be summarized by the following fourteen (S1-S14) states:

S1 A subscriber goes 'off-hook'. The associated LI performs the task T1. The LI sends a message to

the CP via the MD. The MD performs the task T10 and the call enters the state S2.

S2 The CP performs the task T2. The CP sends a message to the SN via the MD. The MD performs the task T11 and the call enters the state S3.

S3 The SN performs the task T3. The SN sends a message to the CP via the MD. The MD performs the task T10 and the call enters the state S4.

S4 The CP performs the task T4. The CP sends a message to the LI via the MD. The MD performs the task T11 and the call enters the state S5.

S5 The LI performs the task T5. The LI sends a message to the CP via the MD. The MD performs the task T10 and the call enters the state S6.

S6 The CP performs the task T6. The CP sends a message to the SN via the MD. The MD performs the task T11 and the call enters the state S7.

S7 The SN performs the task T3. The SN sends a message to the CP via the MD. The MD performs the task T10 and the call enters the state S8.

S8 The CP performs the task T4. The CP sends a message to The SN via the MD. The MD performs the task T11 and the call enters the state S9.

S9 The SN performs the task T7. The SN sends a message to the CP via the MD. The MD performs the task T10 and the call enters the state S10.

S10 The CP performs the task T4. The CP sends a message to the LI via the MD. The MD performs the task T11 and the call enters the state S11.

S11 The LI performs the task T8. The LI sends a message to the CP via the MD. The MD performs the task T10 and the call enters the state S12.

S12 The CP performs the task T4. The CP sends a message to the LI via the MD. The MD performs the task T11 and the call enters the state S13.

S13 The LI performs the task T9. The LI sends a message to the CP via the MD. The MD performs the task T10 and the call enters the state S14.

S14 The CP performs the task T4. The call is now in 'talking' state.

The description given for the call setup process can be summarized in a compact form as a table as shown in Figure 6.

SN	Associated Task		Next Task		Message Flow
	Task	Sub System	Task	Sub System	
S1	T1	LI	T2	CP	LI->MD->CP (T10)
S2	T2	CP	T3	SN	CP->MD->SN (T11)
S3	T3	SN	T4	CP	SN->MD->CP (T10)
S4	T4	CP	T5	LI	CP->MD->LI (T11)
S5	T5	LI	T6	CP	LI->MD->CP (T10)
S6	T6	CP	T3	SN	CP->MD->SN (T11)
S7	T3	SN	T4	CP	SN->MD->CP (T10)
S8	T4	CP	T7	SN	CP->MD->SN (T11)
S9	T7	SN	T4	CP	SN->MD->CP (T10)
S10	T4	CP	T8	LI	CP->MD->LI (T11)
S11	T8	LI	T4	CP	LI->MD->CP (T10)
S12	T4	CP	T9	LI	CP->MD->LI (T11)
S13	T9	LI	T4	CP	LI->MD->CP (T10)
S14	T4	CP	—	—	—

SN – State Number

-> – Direction of flow of message

Figure 6. Routing Table for Architecture-A

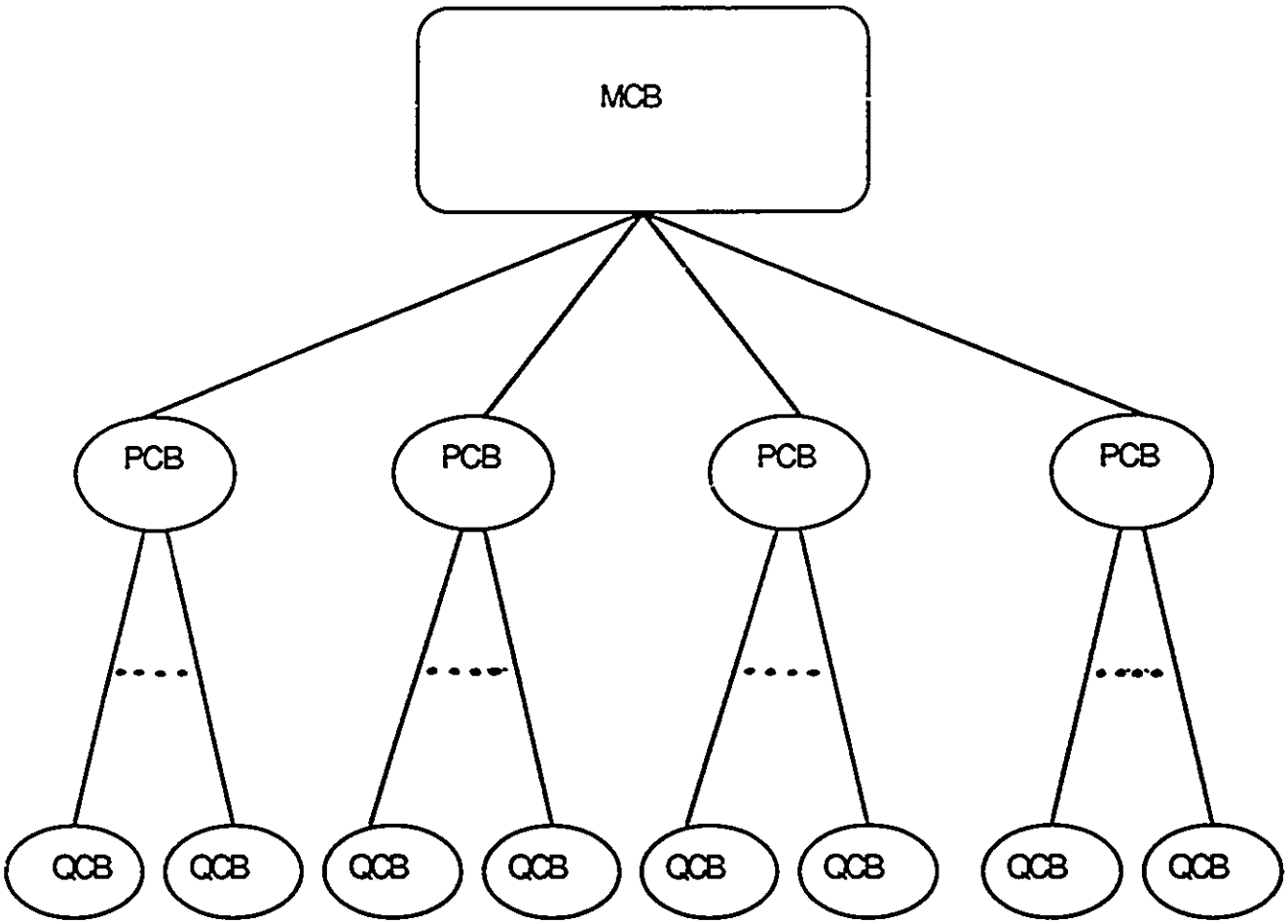
3.2.5 Simulator

The simulator is based on the methodology given in [3]. It consists of two separate programs written in 'C'. The first program (INPUT) consists of about 140 source lines. It is an interactive program that takes input from the terminal and writes it into a file. The contents of the file are input parameters to the second program. The second program (SIMMDSS) consists of about 1400 source lines. It reads data from the file written by program INPUT and simulates call setup in the queueing network. The simulator has been executed on both a VAX 11/750 and an IBM VM/370. The following subsections describe the data structures and the logic used in the simulator.

3.2.5.1 Data Structures

Figure 7 presents a high-level view of the data structures used by the simulator. The data is structured in a hierarchical fashion and has three levels.

MCB The MCB is the Master Control Block and is at the top level in the hierarchy. It contains



MCB - Master Control Block
PCB - Processor Control Block
QCB - Queue Control Block

Figure 7. Data Structures

global variables and constants. The MCB has a list of pointers pointing to the control blocks for the four processors in the queueing network. There is a double linked list called 'event_list' maintained in the MCB that consists of event elements where each such element represents an instance of a task being processed at present.

PCB The PCB is the Processor Control Block. There is one PCB for each processor in the queueing network and is accessible by a pointer maintained in the MCB. The PCB contains processor specific information such as processor_number (number assigned to the processor in the queueing network), number_of_queues_served (number of tasks performed by the processor) and proc_discipline (queue processing strategy) etc..

QCB The QCB is the Queue Control Block. There is one QCB for each queue in the queueing network. The QCB's for queues belonging to a processor are accessed by pointers maintained in the PCB for that processor. The QCB contains queue specific information such as queue_number (number assigned to the queue) and mean_service_time (average time required to perform the task) etc..

The QCB uses a double linked list to queue messages on First-Come-First-Served (FCFS) basis.

3.2.5.2 INPUT Program

The INPUT program is an interactive program that prompts for the values for the parameters required by SIMMDSS. The values entered are written into a file which can be checked or edited if required. Part of the input activity is concerned with entering the data from the Routing Table for the MDSS under consideration. The overall logic flow for INPUT program is given below:

```
OBTAIN number_of_states;
FOR (each state) DO
    BEGIN
    OBTAIN task_number;
    OBTAIN processor_number;
    END
```

From the above input, the program computes number of tasks assigned to each individual processor type.
Each task assigned to a processor type is represented by a queue served by that processor in the queueing network.

```
FOR (each processor) DO
    BEGIN
    OBTAIN number_of_servers;
    OBTAIN proc_discipline;
    FOR (each task assigned to the processor) DO
        BEGIN
        OBTAIN mean_service_time;
        OBTAIN priority_attribute;
        END
    END
```

From the above input, number_of_servers is used to vary the processing power of the subsystem. For example, by increasing the number_of_servers from one to two, the processing power of the processor can be doubled.
The proc_discipline specifies the scheduling algorithm to be used for processing the queues. The expected input is FCFS or LCFS or PS. In our model, only FCFS is used.
The mean_service_time is associated with individual task and it gives the time required to complete the task.
The priority_attribute identifies a priority queue in the queueing network. The expected input is Yes/No.

```
OBTAIN number_of_calls;
OBTAIN number_of_events;
```

Number_of_calls specifies the number of calls to be injected into the queueing network.

Number_of_events specifies the length of the simulation run.

3.2.5.3 SIMMDSS Program

The SIMMDSS program is the program that simulates call setup in the queueing network. The input parameters required by this program are obtained from the file created by INPUT program. The program reads the file, initializes the variables and executes for a specified number of events. At the end of the simulation run, the program computes and writes performance estimates in a file.

There are three major routines in SIMMDSS that are described below:

Main — It defines the data structures and variables used in this program. It reads input parameters from the file created by INPUT program, sets up control blocks and initializes variables. The input parameters describe the queueing network character-

istics. The calls are represented by the messages circulating in the queueing network. To begin with, all the messages are present in the task queue representing the task 'Detect Off-Hook' at node LI (refer to figure 4, 5 and 6) and are scheduled for processing subject to the availability of the processor. The program recognizes task completion as the only event. It maintains a double linked list (event_list) of the event elements ordered by the time of occurrence, each representing a task currently being performed by one of the four processors. During initialization, the event elements are inserted into the list for those messages that are scheduled for processing. After the initialization is done, the routine execute a LOOP for a specified number of events that control the length of the simulation run. For each event in the list, routine Processor_control is called with the action code 'Insert' or 'Remove'. The action code 'Insert' is used when a message is to join a queue at a processor. The action code 'Remove' is used when a message finishes processing and has to

leave the current processor to go to the next processor. At the end of the simulation run, performance estimates are computed from the statistics collected during the run and are written into a file.

Processor_Control The routine Processor_Control maintains statistics for each processor in the queueing network. It is responsible for scheduling messages waiting in the queues for processing whenever a processor becomes free. It calls Queue_Control with the same action code (i.e. 'Insert' or 'Remove') passed onto it by routine Main.

Queue_Control The routine Queue_Control maintains statistics for each task queue in the network. Based on the action code, it either inserts or removes a message from the queue.

The overall logic flow for SIMMDSS program is given below:

```
PROCEDURE MAIN;
  BEGIN
  Define data structures (MCB,PCB,QCB);
```

```
  Define variables (clock, events_done);
  Read   input_file;
  Initialize control_structures (MCB,PCB,QCB);
  Initialize Routing_table, event_list;
  LOOP
    Remove an event from the event_list;
    Update clock;
    Setup appropriate PCB;
    Call Processor_control (remove);
    From the Routing_Table, find the next processor and queue;
    Call Processor_Control (insert);
    Increment events_done;
  UNTIL events_done < number_of_events;
  Compute performance estimates;
  Write performance estimates;
  END

PROCEDURE PROCESSOR_CONTROL(action_code);
  BEGIN
  Update processor statistics;
  IF (action_code = remove) THEN
     BEGIN
     Setup appropriate QCB;
     Call Queue_Control(remove);
     IF (messages waiting in the queues) THEN
        BEGIN
        Schedule next message according to given discipline;
```

```
        Insert event in the event_list;
        END
     END
  ELSE
     BEGIN
     Setup appropriate QCB;
     Call Queue_control(insert);
     IF (processor is free) THEN
        BEGIN
        Schedule new message for processing;
        Insert event in the event_list;
        END
     END
  END

PROCEDURE QUEUE_CONTROL(action_code);
  BEGIN
  Update queue statistics;
  IF (action_code = remove) THEN
     BEGIN
     Remove message from the queue;
     Decrement queue_length;
     END
  ELSE
     BEGIN
     Insert message in the queue;
     Increment queue_length;
```

```
    END
END
```

3.2.5.4 Output from SIMMDSS

The output from the simulator includes:

1. Processor utilization of the processors. This measures the fraction of the time each processor is busy and is presented as a percentage.
2. Processor throughput of the processors. This is the number of the messages processed per unit time and is equal to the service rate times the fraction of time the processor is busy.
3. Average Processing Time for each message. This includes the queueing delay caused because of the contention for resources.
4. Average queueing length at each processor. This is the average number of messages waiting to be served. The minimum and maximum queue length is also noted.
5. The total number of calls setup during the simulation run.

6. Call-throughput for the system. This is a measure of the number of calls that were setup in a unit time.
7. Call-attempts per hour. This is the maximum number of calls that can be handled by the system in one hour.
8. Average Call-setup time. This is the average time required to setup a call from beginning (off-hook) to end excluding the time taken by the subscriber to dial in the digits.
9. Average Call-delay time. This is the delay introduced in the call setup time due to contention for resources. It is equal to the sum of all the queueing delays that the call encounters during its progress.

CHAPTER 4

VALIDATION AND RESULTS

This chapter presents a validation for the performance model described in the previous chapter. It also includes a summary and the analysis of the experimental results for the MDSS for different loads. The results are used to predict the dynamic behavior of the MDSS.

4.1 Validation

The validity of the queueing network model is examined by comparing the values of the performance measures of the model with the data available from the literature for existing MDSS's [5,20]. Both the architecture and the software structure for these systems have already been described in chapter 2.

The system [5] is composed of up to seven Telephony Processors (TP's), a Message Distributor Circuit (MDC), a

Space and Time switch and up to sixty four Peripheral Processors (PP's). The TP's perform analysis and sequencing functions of the call processing and operate in load sharing mode. However, the processing of a call is handled by a TP. The PP's interface with physical lines and Space and Time switch provides physical connections for a call. The TP's and PP's communicate with each other through MDC. The distribution of call processing functions (telephony tasks) for this system has already been described in chapter 3. In the queueing network model, the TP's, the MDC, the Space and Time Switch and the PP's are represented by the CP, the MD, the SN and the LI respectively. The effect of multiple TP's is taken into account by changing the processing power of the CP.

The performance of any MDSS is obtained by performing extensive experiments on the real system. The MDSS is subjected to heavy loads by connecting it to a source capable of generating a large number of call such that there is always a call request waiting to be served as long as the system has free resources and capacity to accept more calls. The experiment is run for a specified period of time and the performance measures consisting of the statistics on call attempts/hour, total CCS load, the call-throughput, the processor utilization, the call-setup time and the call-delay time are obtained.

Similar experiment is conducted with the queueing network model. The simulation program behaves in a way such that whenever the LI is free and the number of calls in the queueing network is less than the total number of calls allowed, another call is introduced. The actual performance measures are used to determine value for parameters in the queueing network model.

The results from the experiment are presented in Figure 8. The processing power of the processors is changed to obtain results for a range of capacities of the MDSS.

Proc-essor Power	Actual Results		Model Results		Percentage Of Relative Error
	Call Attempts/ Hour	Proc. Utiliz. (%)	Call Attempts/ Hour	Proc. Utiliz. (%)	
1	120,000	90.0	114,000	90.4	- 5.00
2	300,000	90.0	273,600	90.8	- 9.13
3	360,000	—	326,000	90.7	- 9.44

Figure 8. Call Attempts/Hour and % of Relative Error

The number of call attempts per hour for the MDSS is measured at 90% processor utilization. The model results are

accurate within 5% for the MDSS to handle 120,000 call-attempts per hour. The relative error in the results increases to 9.5% as the processing power is increased to handle 360,000 call-attempts per hour.

4.2 Results

A number of experiments were conducted to determine the processor utilization, the average queueing delay on a per task basis, the average call-setup time, the average call-delay time and the call-throughput for the MDSS. The results obtained from the experiments along with the analysis are presented below.

An interesting case is when there is one call in the system. The results obtained are given in Figure 9.

Call-setup Time	Call-delay Time	Call-throughput
0.134	0.000	7.481

Figure 9. Performance Estimates with One Call

With only one call in the system, there is no contention for resources. The call does not have to queue at any time during its progress from one state to another state. There are no queueing delays and hence, the average call-delay time is zero. The call-setup time is equal to the actual time required for executing the telephony tasks.

It should be noted here that the values for the call-setup time, the call-delay time and the call-throughput given in Figure 9 represent lower bounds on the call-setup time, the call-delay time and the call-throughput time respectively for the MDSS.

To determine the behavior of the system under different loads, the number of calls in the system was increased. The average call-setup time, the average call-delay time and the

call-throughput were noted. The results obtained are summarized in Figure 10 to Figure 13.

As the number of calls is increased, both the processor utilization and the contention for resources increase resulting in increase in the queueing delays. The increase in the processor utilization and the average queueing delay for the number of calls in the range of 1-360 is presented in Figure 10 and in Figure 11 respectively.

Number Of Calls	Processor Utilization			
	CP	MD	SN	LI
1	24.6	25.1	25.2	25.1
2	35.0	36.2	35.6	35.8
4	46.6	47.7	47.1	47.4
8	59.5	58.4	57.4	58.4
15	67.4	67.4	66.4	67.4
30	75.8	75.6	76.0	74.9
60	81.9	81.8	82.1	82.0
90	85.0	85.0	84.9	85.0
120	86.8	87.0	86.8	86.9
150	88.0	87.9	88.3	88.3
180	89.3	89.1	89.4	89.1
210	89.8	89.7	89.9	89.8
240	90.7	90.4	90.4	90.4
270	91.0	90.9	91.1	91.2
300	91.0	91.2	91.4	91.6
360	92.1	92.2	92.1	92.1

Figure 10. Number of Calls and Processor Utilization

Number Of Calls	Average Queueing Delay Per Task			
	CP	MD	SN	LI
1	0.000	0.000	0.000	0.000
2	0.006	0.003	0.013	0.017
4	0.007	0.004	0.016	0.034
8	0.009	0.005	0.020	0.067
15	0.012	0.006	0.025	0.125
30	0.016	0.009	0.036	0.248
60	0.022	0.012	0.046	0.501
90	0.027	0.014	0.053	0.750
120	0.030	0.016	0.061	1.000
150	0.034	0.017	0.070	1.254
180	0.038	0.019	0.075	1.499
210	0.041	0.021	0.083	1.751
240	0.045	0.022	0.086	1.999
270	0.045	0.023	0.090	2.255
300	0.046	0.023	0.096	2.510
360	0.052	0.027	0.107	3.000

Figure 11. Number of Calls and Average Queueing Delay

The processor utilization and the average queueing delay is proportional to the number of calls in the system. The processor utilization and the average queueing delay increases with increase in the number of calls. The average queueing delay is very small for number of calls up to 60 but becomes quite considerable after that. We will see later how it affects the call-setup time and the capacity of the MDSS.

For the same range of number of calls in the system, we observe the average call-setup time and the average call-delay time. The average call-setup time is equal to the sum of actual time required for executing telephony tasks and the average call-delay time. The average call-delay time is the sum of queueing delays encountered during the progress of the call. The results obtained are presented in Figure 12.

Processor Utilization				Call Setup Time	Call Delay Time
CP	MD	SN	LI		
24.6	25.1	25.2	25.1	0.134	0.000
35.0	36.2	35.6	35.8	0.188	0.052
46.6	47.7	47.1	47.4	0.281	0.141
59.0	58.4	57.4	58.4	0.460	0.319
67.4	67.4	66.4	67.4	0.746	0.600
75.8	75.6	76.0	74.9	1.332	1.182
81.9	81.8	82.1	82.0	2.426	2.278
85.0	85.0	84.9	85.0	3.608	3.240
86.8	87.0	86.8	86.9	4.393	4.118
88.0	87.9	88.3	88.3	6.004	5.470
89.3	89.1	89.4	89.1	6.785	6.312
89.8	89.7	89.9	89.8	7.969	7.428
90.7	90.4	90.4	90.4	8.690	8.338
91.0	90.9	91.1	91.2	9.820	9.418
91.0	91.2	91.4	91.6	10.613	10.358
92.1	92.2	92.1	92.1	13.936	13.568

Figure 12. Call-setup and Call-delay Time

Once again, some interesting results are obtained. Results show that both the average call-setup time and the average call-delay time is very small for processor utilization below 67%. However, both increase very sharply as the number of calls is increased and the processor utilization is in the range of 67-90%. From the definition of the average call-setup time and the average call-delay time and the behavior of the queueing delay observed previously, these are the expected results. When there is small number of calls in the system, contention for resources is very small and hence, the queueing delays are negligible. The net effect is that the average call-delay time is small which results in small call-setup time. When the number of calls is increased beyond certain point, the contribution of the queueing delay becomes quite considerable resulting in large average call-setup time.

Figure 13 presents results for the call-throughput and the call-attempts per hour. By definition, the call-throughput is the number of calls setup in unit time and depends on the utilization of the resources and the service rate. If the service rate is kept constant, then the call-throughput is proportional to the processor utilization or the number of calls in the system. Results show that the call-throughput increases with increase in the number of calls in the system.

Processor Utilization				Call Through-put	Call Attempts per Hour
CP	MD	SN	LI		
24.6	25.1	25.2	25.1	7.481	25200
35.0	36.2	35.6	35.8	10.696	36000
46.6	47.7	47.1	47.4	14.145	50400
59.0	58.4	57.4	58.4	17.511	61200
67.4	67.4	66.4	67.4	20.152	72000
75.8	75.6	76.0	74.9	22.660	79200
81.9	81.8	82.1	82.0	24.540	86400
85.0	85.0	84.9	85.0	25.453	90000
86.8	87.0	86.8	86.9	26.021	93600
88.0	87.9	88.3	88.3	26.343	93600
89.3	89.1	89.4	89.1	26.674	93600
89.8	89.7	89.9	89.8	26.854	93600
90.7	90.4	90.4	90.4	27.072	97200
91.0	90.9	91.1	91.2	27.216	97200
91.0	91.2	91.4	91.3	27.291	97200
92.1	92.2	92.1	92.1	27.530	97200

Figure 13. Call-throughput and Call-attempts/hour

Call-attempts per hour also increases with increase in number of calls in the system. Call-attempts per hour is bounded by the call-setup time.

The results obtained above are very useful in determining the dynamic behavior of a MDSS under different loads. The experiment conducted with one call produces the lower bounds for the average call-setup time, average call-delay time and call-throughput. Experiments with large number of calls produces upper bounds on the average call-setup time.

average call-delay time, call-throughput and call-attempts per hour. Also, these results can be used to determine the capacity of the MDSS. Corresponding to the maximum number of calls that can be handled by the system, maximum number of lines that can be supported by the system can be determined.

The results obtained from the experiments show that the proposed model is justified to be used as a tool during design stages of MDSS. It can also be used to study the impact of design changes and introduction of new features on the overall performance of the system.

CHAPTER 5

A COMPARATIVE ANALYSIS OF TWO MDSS'S

In this chapter the model developed is used to study the effects of system architecture on the performance of an MDSS. The Architecture-A and the Architecture-B described in Chapter 2 are analyzed and their performance are compared.

The system architecture is the result of the interconnection arrangement used between the subsystems of an MDSS and is one of the crucial factor affecting the system performance. The flow of messages in the system depends on the message links provided between the various subsystems. In our model, the architecture of the system and the resulting flow of messages is reflected by the Routing Table (RT). The difference between the two architectures are described below.

Architecture-A: The SN and the LI's are directly connected to the MD. Separate links are used to connect the SN and each LI to the MD. The MD has routing information for both the SN and the LI's. The MD receives messages from

the CP, determines its destination and routes it on a appropriate link to the SN or the LI. Messages from the SN and the LI's are received on separate link and are routed to the CP.

Architecture-B: The four subsystems are connected in linear fashion. It differs from Architecture-A the way the SN and the LI's are connected to the MD. The SN is connected to the MD by a link. Each LI is connected to the SN by a link. In this architecture, the MD does not know about the physical distribution of the LI's, it routes messages from the CP to the SN and from the SN to the CP. When the SN receives a message from the MD, it determines its destination. If the message is meant for the SN, it is processed locally, otherwise it is routed on appropriate link to the LI. The messages received by the SN from the LI's are forwarded to the MD.

5.1 Assumptions for Comparison

The comparison of performance for the two architectures is subject to the following assumptions:

- The two systems are made up of identical hardware, i.e. the processing power of the corresponding subsystems is same.
- The two systems have identical software structure, i.e. the distribution of telephony tasks between the subsystems is same for both the systems.
- In both the systems, the number of distinct processing steps required to setup the call is same. It means that the Routing Table for call setup in both the systems has same number of entries.

5.2 RT for the two Architectures

The RT for the Architecture-A is described in Chapter 2 (Figure 6 on page 39). The distribution of telephony tasks and the RT for the Architecture-B is shown in Figure 14 and Figure 15 respectively.

Task Number	Task Assigned To Subsystem			
	CP	MD	SN	LI
T1				X
T2	X			
T3			X	
T4	X			
T5				X
T6	X			
T7			X	
T8				X
T9				X
T10		X		
T11		X		
T12			X	
T13			X	

Figure 14. Distribution of the Telephony Tasks for Architecture-B

SN	Associated Task		Next Task		Message Flow
	Task	Sub System	Task	Sub System	
S1	T1	LI	T2	CP	LI->SN->MD->CP (T12,T10)
S2	T2	CP	T3	SN	CP->MD->SN (T11)
S3	T3	SN	T4	CP	SN->MD->CP (T10)
S4	T4	CP	T5	LI	CP->MD->SN->LI (T11,T13)
S5	T5	LI	T6	CP	LI->SN->MD->CP (T12,T10)
S6	T6	CP	T3	SN	CP->MD->SN (T11)
S7	T3	SN	T4	CP	SN->MD->CP (T10)
S8	T4	CP	T7	SN	CP->MD->SN (T11)
S9	T7	SN	T4	CP	SN->MD->CP (T10)
S10	T4	CP	T8	LI	CP->MD->SN->LI (T11,T13)
S11	T8	LI	T4	CP	LI->SN->MD->CP (T12,T10)
S12	T4	CP	T9	LI	CP->MD->SN->LI (T11,T13)
S13	T9	LI	T4	CP	LI->SN->MD->CP (T12,T10)
S14	T4	CP	—	—	—

SN – State Number

-> – Direction of flow of message

Figure 15. Routing Table for Architecture-B

In Figure 14, the SN performs two additional tasks. It receives the messages from the MD and routes them to the appropriate LI (this is task T12) and forwards the messages received from the LI's to the MD (this is task T13). Note that both the RT's have same number of entries and the telephony task associated with each entry is identical. The only difference is in the flow of the messages between the CP and the LI's.

5.3 Comparative Analysis

A number of experiments were conducted for each of the two architectures to obtain the processor utilization of the CP, the average call-setup time, the average call-delay time and call-throughput over a range of system load. The system load was changed by varying the number of calls in the system. An analysis of the results obtained is given below.

5.3.1 Processor Utilization of the CP

The behavior of the utilization of the CP's for the two architectures is shown in Figure 16. The utilization of the CP rises sharply with increase in the number of calls in the system. For Architecture-A, it reaches 85% with 90 calls in the system. However, for Architecture-B, the utilization of the CP never rises above 65%. From the analysis of the results obtained for the Architecture-B, it was observed that when there are more than 60 calls in the system, the SN becomes a bottleneck. The average queue length and hence the average queueing delay becomes high. This results in under utilization of the CP and the other subsystems.

5.3.2 Call-setup and Call-delay Time

Figure 17 shows call-setup time as the function of the number of calls in the system for the two architectures. When there are less than 100 calls in the system, the system architecture has very little effect on the call-setup time. The call-setup time for the two architectures is almost same. The results is due to the fact that, under light loads, the average queueing delay is very low, making the

presence of additional queueing delay, if any, because of a specific architecture negligible.

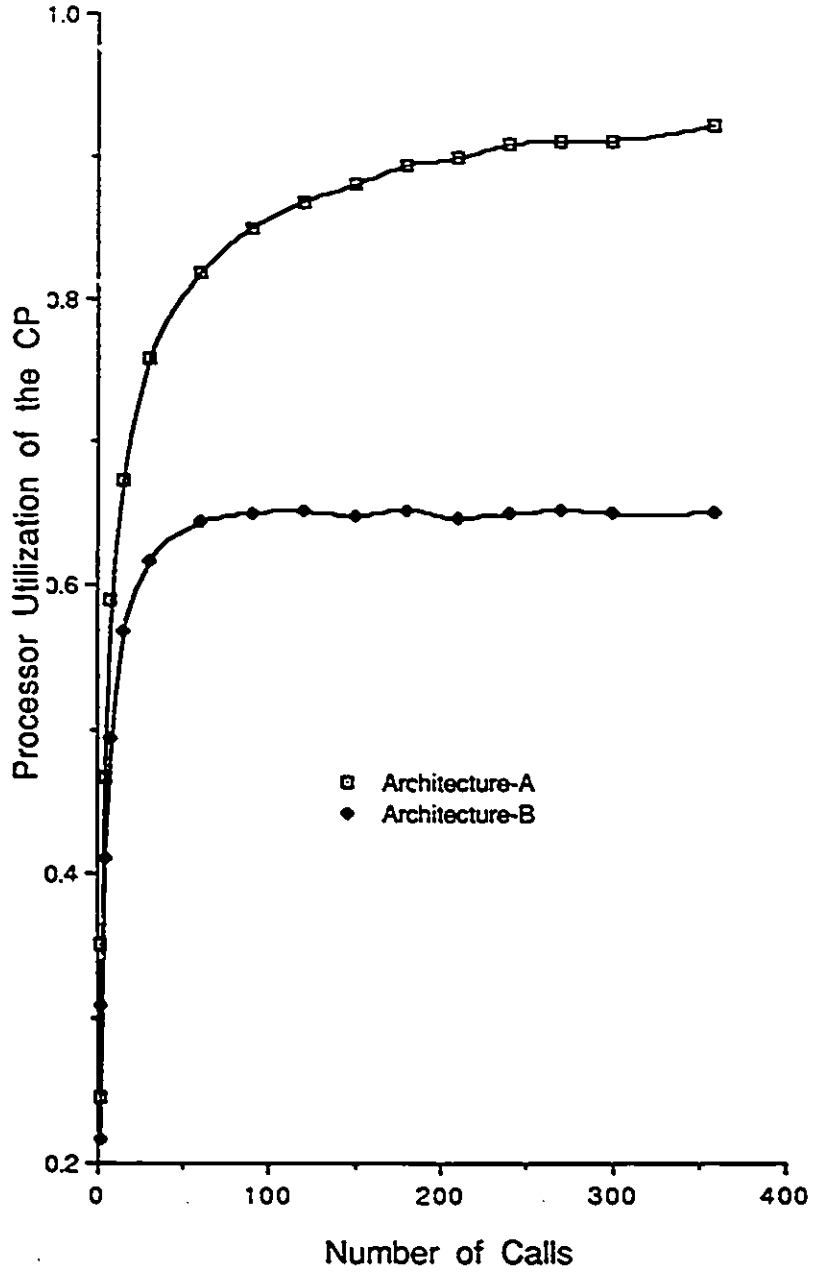


Figure 16. Comparison of Processor Utilization

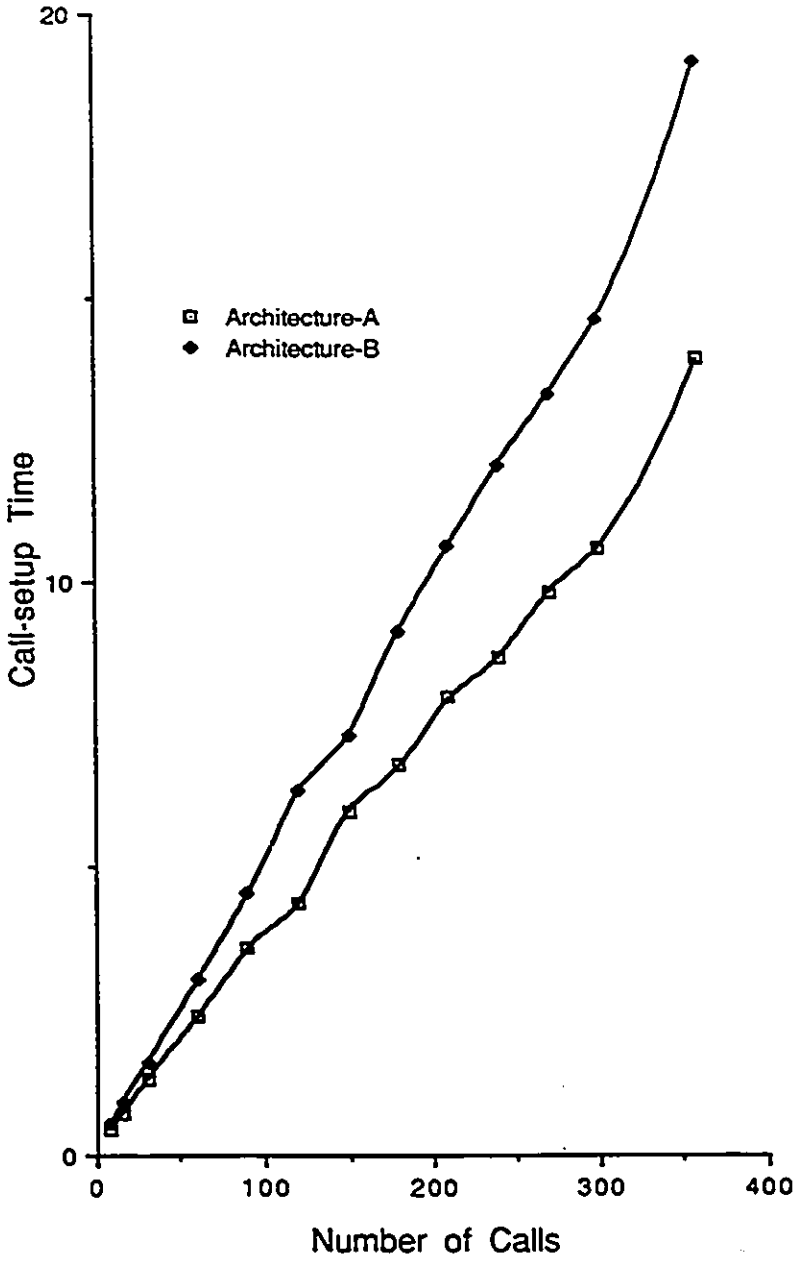


Figure 17. Comparison of Call-setup Time

But as the number of calls is increased, the effect of the system architecture becomes clearly visible. For number of calls in the range of 150-360, the call-setup time for Architecture-B rises very sharply. For this range of load, Architecture-A is certainly superior to Architecture-B as it gives lower call-setup time as compare to the Architecture-B. The behavior can be explained by the fact that the Architecture-B introduces additional queueing delay overhead because of extra hops for the messages exchanged between the CP and the LI. An analysis of the queues at the tasks level indicates that the average queueing delay at the SN for the Architecture-B is larger than the average queueing delay at the SN for the Architecture-A.

In Figure 18 the queueing delay is plotted as the function of the number of calls in the system. The slopes of the curves for the queueing delay are similar to that of the call-setup time. This is consistent with what was observed in the previous chapter. The call-setup time remains almost constant for the entire range of loads, it is the call-delay time that changes with the change in number of calls in the system.

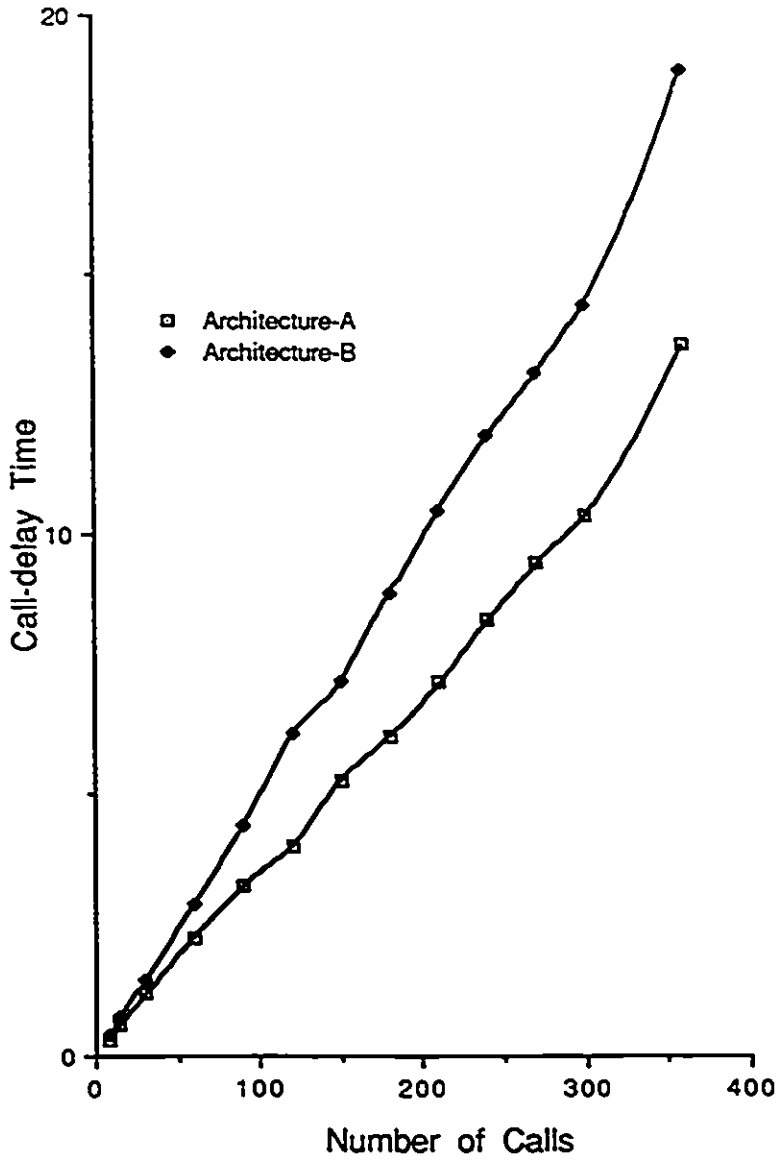


Figure 18. Comparison of Call-delay Time

5.3.3 Call-throughput

Figure 19 shows the call-throughput as the function of the number of calls in the system for the two architectures. The call-throughput for the Architecture-A is higher than the call-throughput for the Architecture-B. The call-throughput of the system depends on the processor utilization and the service rate. We have seen before that in case of the Architecture-A, the utilization of the CP and the LI's is severely restricted by the SN. This results in lower call-throughput from the system.

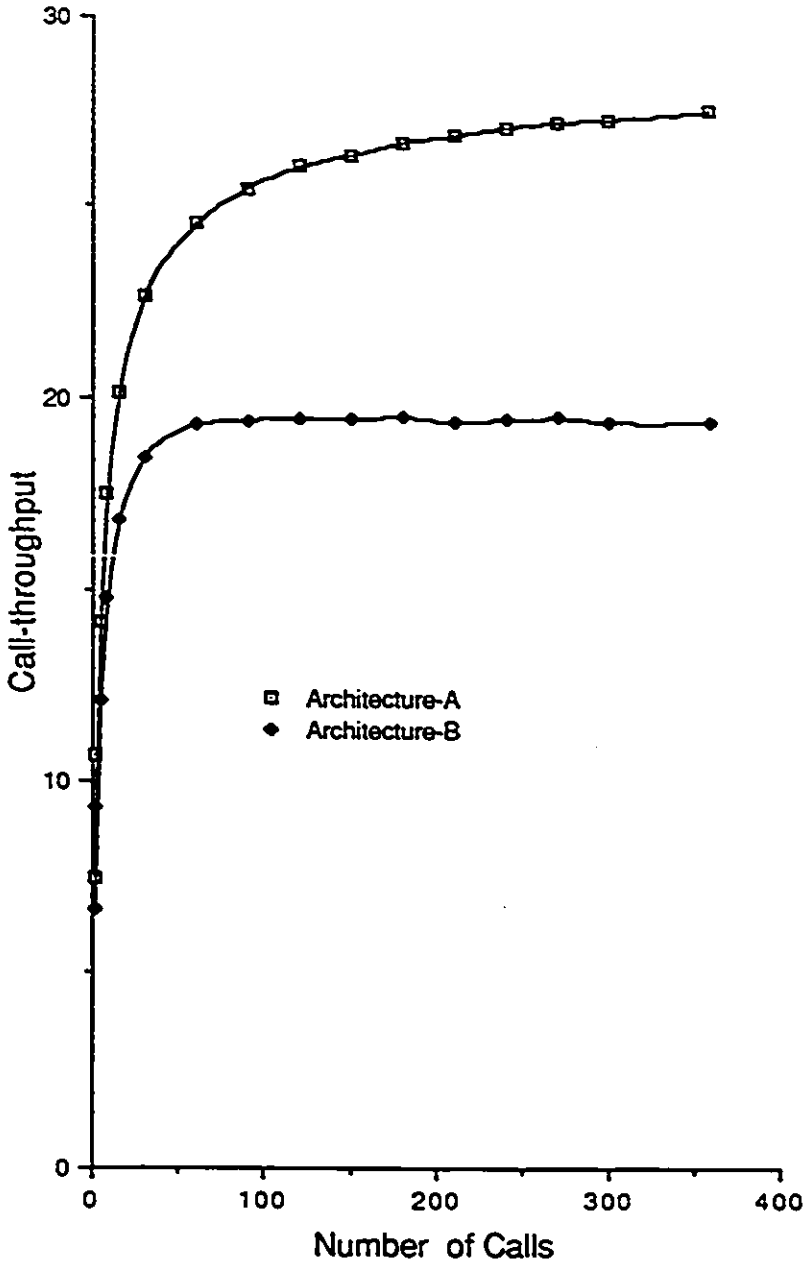


Figure 19. Comparison of Call-throughput

From the results, we arrive at a conclusion that with small number of calls in the system, the architecture has no significant impact on the performance of the system. However, as the number of the calls is increased, the Architecture-A clearly outperforms the Architecture-B in terms of the call-setup time and the call-throughput. The Architecture-A gives lower call-setup time and yields higher call-throughput. These results are also supported by the literature.

By conducting more experiments it was observed that the performance of the Architecture-B can be improved by increasing the processing power of the subsystem SN. The experiment results showed that if the processing power of the SN is increased by one and a half times then the Architecture-B performs equally well as the Architecture-A.

CHAPTER 6

SUMMARY, CONCLUSIONS AND FUTURE WORK

6.1 Summary

A study of several existing multiprocessor digital switching systems revealed similarities in their structural organization and operation principles and it served as a motivation to develop a model for studying the relative performance of such systems.

A closed queueing network model representing the structure of a class of multiprocessor digital switching systems was proposed. Attention was focused on the call setup process within an MDSS. A call was represented as a job in the queueing network and was characterized by the telephony tasks and the subscriber events. A simulation methodology relating to the performance determination during the call setup process developed and the performance measures such as the processor utilization, the average call-setup time, the

average call-delay time, the call-throughput and call-attempts per hour were obtained.

The validity of the model was established by tailoring it to one particular existing system and comparing its behavior with the available data for the existing system. The results provided lower and upper bounds for the average call-setup time, the average call-delay time, the call-throughput and the call-attempts per hour for an MDSS. The model was also used to study the impact of the system architecture on the performance of an MDSS.

6.2 Conclusions

The performance evaluation of an MDSS involves very complex, expensive and time consuming experimentation with actual hardware. Furthermore, if the system does not meet the predetermined performance criteria or it exhibits problem in the operations, then it takes tremendous amounts of effort in terms of design changes to correct the problem.

The performance model proposed in this thesis can provide a very cost-effective and accurate solution for the above mentioned problem. The results obtained from the model

justify its use as a tool for understanding and predicting both the dynamic behavior and the likely performance characteristics during the design phases of an MDSS. It can also be used to study the impact of both the design changes and the addition of new features in the existing MDSS's.

By comparing the performance of two specific MDSS's, we have also justified the role of the model during the design selection process.

Some work that is not included in this thesis involved experimentations with the model using several queue processing strategies such as assigning priority to the queues and processing them according to the priority, processing the current queue to its end, processing only the active messages in the queue (i.e. when a queue is started, the messages present in the queue are counted and processed. The message joining the queue during processing are processed in the next turn) and round-robin (i.e. one message from each queue). These different strategies did not show any significant change in the results because of the closed nature of the queueing network and the state dependent routing.

6.3 Future Work

The model proposed in this thesis can be extended in two ways:

1. By including the phase 2 (talking state) and the phase 3 (call disconnect) of the call processing functions. An other node can be added to represent calls in the phase 2. The introduction of these phases may require partitioning of the messages into separate classes in the queueing network.

2. By developing a similar but open queueing network model. Once again, an extra node can be included to model calls in the phase 2. The arrival of the calls can be modeled by a source node and the arrival rate for the calls can be used as a parameter to vary the load in the system. When the call is disconnected, it can be taken out of the queueing network by means of a sink node.

From both the above mentioned approaches, the call-attempts per hour can be obtained by zeroing or eliminating processing in the phase 2.

REFERENCES

1. Kleinrock, L., "Queueing Systems, Volume I: Theory", Wiley, New York (1975).

2. Kleinrock, L., "Queueing Systems, Volume II: Computer Applications", Wiley, New York (1976).

3. Sauer C.H., Chandy K.M., "Computer Systems Performance Modeling", Prentice-Hall, New Jersy (1981).

4. Bell-Northern Research, "Telesis", Special issue DMS-100 family of digital switches, Volume 7 Number 4 (1980).

5. Mnichowicz D.A., Zelinski P.A., "GTD-5 EAX - An Introduction", GTE Automatic Electric Journal, March-April 1982, pp. 46-49.

6. Blondeau E.E., Conlon M.E., Gentzler G.L., "Real-Time Simulation Modeling of the GTD-5 EAX" ISS' 81 CIC Montreal, 1981.

7. Jackson D., Patfield K., "Impacts of Multiprocessing on GTD-5 EAX Call Processing and Operating System Soft-

ware", GTE Automatic Electric Journal, March-April 1982, pp. 50-56.

8. Ballard M., Martin M., Randall T., Renault J.P., "The E10.S-TSS.5 : A Multipurpose Digital Switching System", ISS' 81 CIC Montreal, 1981.

9. Davis J.H., Janik J., Royer R.D., Yokelson B.J., "No. 5 ESS System Architecture", ISS' 81 CIC Montreal 1981.

10. Sauer C.H., MacNair E.A., "Simulation of Communication systems", Prentice-Hall, New Jersy (1982).

11. Marsan M.A., Gerla M., "Markov Models for Multiple Bus Multiprocessor Systems", IEEE Transactions On Computers, Vol. C-31, No. 3, March 1982, pp. 239-248.

12. Kriz J., "A Queueing Analysis of a Symmetric Multiprocessor with Shared Memories and Buses", IEEE Proc., Vol. 130, Pt. E, No. 3, May 1983, pp. 83-89.

13. Chiu W.W., Dumont D., Wood R., "Performance Analysis of a Multiprogrammed Computer System", IBM Journal of Research and Development 19, 3 May 1975, pp. 263-271

14. Fletcher G.Y., Perros H.G., Stewart W.J., "A Queueing Network Model of a Circuit Switching Access Scheme in an

Integerated Services Environment", IEEE Transactions On Communications. Vol. COM-34, No. 1, January 1986, pp. 25-30

15. Beilner H., "On The Construction of Computing System Simulators", Experimental Computer Performance and Evaluation, D. Ferrari, M. Spadoni, Eds., North-Holland Publishing Company (1981), pp. 1-29.

16. Woodbury M.H., Shin K.G., "Performance Modeling and Measurement of Real-Time Multiprocessors with Time-Shared Buses", IEEE Transactions On Computers, Vol. 37, No. 2, February 1988, pp. 214-224.

17. Kobayashi H., Chandy K.M., Yeh R.T. "System Design and Performance Analysis using Analytic Models".

18. C.M. Krishna, K.G. Shin "Performance Measures for Multiprocessor Controllers", Performance '83, A.K. Agrawala and S.K. Tripathi, Eds. New York: North-Holland

19. C.H. Sauer, E.A. MacNair, J.F. Kurose "Queueing Network Simulation Of Computer Communication", IEEE Journal On Selected Areas In Communications, Vol. SAC-2, No.1, January 1984

20. J.J. Otter, B.K.Penney, "DMS-100 System Stability And Maintainability Under Heavy Traffic", National Telecommunications Conference, November 1981.

21. B.K. Penney, J.W.J. Williams, "The Software Architecture For A Large Telephone Switch", IEEE Transcations On Communications, Vol. COM-30, No. 6, June 1982, pp. 1369-1378.