NOTICE

The quality of this microfiche 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.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

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

THIS DISSERTATION HAS BEEN MICROFILMED EXACTLY AS RECEIVED

AVIS

La qualité de cette microfiche 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.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, examens publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de ce microfilm est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30.

LA THÈSE A ÉTÉ MICROFILMÉE TELLE QUE NOUS L'AVONS RECEUE
Performance Modeling of a
Loosely coupled Multi-microcomputer System

D. Muralidharan

A Thesis
in
The Department
of
Computer Science

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Computer Science at
Concordia University
Montréal, Québec, Canada.

August 1985

© D. Muralidharan, 1985
Permission has been granted to the National Library of Canada to microfilm this thesis and to lend or sell copies of the film.

The author (copyright owner) has reserved other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without his/her written permission.

L’autorisation a été accordée à la Bibliothèque nationale du Canada de microfilmer cette thèse et de prêter ou de vendre des exemplaires du film.

L’auteur (titulaire du droit d’auteur) se réserve les autres droits de publication; ni la thèse ni de longsextraits de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation écrite.

ABSTRACT

Performance Modeling of a loosely coupled Multi-microcomputer system.

D. Muralidharan

A loosely coupled Multi-microcomputer system comprises of autonomous, multiple microcomputers which interact with each other, by passing messages over a communication network. Such a system can be designed to provide higher processing power or better system flexibility and reliability in contrast to a single microcomputer system.

CUENET (Concordia University Educational NETwork) is one such multi-microcomputer system consisting of both 8 and 16 bit microcomputers, arranged in a master-slave hierarchy. They communicate with each other by passing messages over a parallel bus called C-Bus under the control of a centralized bus controller.

The performance of a multi-microcomputer system, besides other factors, is influenced by the manner in which the processes running on the individual computers interact. In performance modeling of multi-microcomputer systems, these aspects are studied using a functional model of the system.
Stochastic Petri nets (SPNs) and Generalized Stochastic Petri nets (GSPNs) have been employed in the performance modeling of multiple computer systems. SPN and GSPN modeling techniques have the ability to describe the concurrency and conflict inherent in the system in parallel processing. Delay and throughput are the two performance measures that are derivable from the analysis of such models. The application of SPN and GSPN modeling techniques, found in the recent publications, are confined to modeling tightly coupled systems.

In this thesis, we have applied the SPN and GSPN modeling techniques to study the performance of CUENET from the message communication point of view. The detailed SPN, GSPN models and aggregate GSPN models for two and three processor CUENET configuration are presented and results of the analysis of these models are discussed. The applicability of the SPN and GSPN techniques to loosely coupled multi-microcomputer systems is demonstrated.
ACKNOWLEDGEMENT.

I would like to express my indebtedness to my supervisor Dr. T. Radhakrishnan. His guidance, assistance and encouragement were an asset during the thesis work, particularly and in my stay at Concordia, in general.

My thanks are due to Dr. V.S. Alagar who gave me valuable advice during the thesis work despite his busy schedule.

I wish to express my sincere and heartfelt thanks to my friends Mr. K. Venkatesh and Mr. R.N. Prasad who have helped me at various stages of my thesis work and especially in bringing the thesis report to the final shape by their drafting skills. Their presence has made my stay in Concordia very enjoyable and memorable.

Dr. M.A. Marsan of Polytechnico de Torino, Torino, Italy has helped me in the thesis work by providing the GSPN software. My thanks are due to him and his colleagues.

My thanks are due to Mr. Clifford Grossner who was always available for consultation regarding CUENET.

I wish to express my sincere thanks to all of my department colleagues and staff who have made my stay at Concordia a very pleasant experience.

My sincere thanks are due to the Canadian Commonwealth Scholarship committee for the financial support during my
graduate studies.

I wish to express my heartfelt thanks to my dear wife for taking care of our family in India during my stay here.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title Page</td>
<td>i</td>
</tr>
<tr>
<td>Signature Page</td>
<td>ii</td>
</tr>
<tr>
<td>Abstract</td>
<td>iii</td>
</tr>
<tr>
<td>Acknowledgement</td>
<td>v</td>
</tr>
<tr>
<td>Table of contents</td>
<td>vii</td>
</tr>
<tr>
<td>List of Figures and tables</td>
<td>ix</td>
</tr>
<tr>
<td>I. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Evaluation of Distributed Computing Systems (DCS)</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Factors in the design of DCS</td>
<td>7</td>
</tr>
<tr>
<td>1.3 Performance Evaluation of DCS</td>
<td>9</td>
</tr>
<tr>
<td>1.4 Thesis motivation and outline</td>
<td>18</td>
</tr>
<tr>
<td>II Modeling Techniques for Performance Analysis of DCSs</td>
<td>21</td>
</tr>
<tr>
<td>2.1 General requirements for modeling DCS</td>
<td>21</td>
</tr>
<tr>
<td>2.2 Petrinets</td>
<td>26</td>
</tr>
<tr>
<td>2.3 Stochastic Petrinets (SPNs)</td>
<td>40</td>
</tr>
<tr>
<td>2.4 Generalized Stochastic Petrinets (GSPNs)</td>
<td>48</td>
</tr>
<tr>
<td>III. The Architecture of CUENET</td>
<td>62</td>
</tr>
<tr>
<td>3.1 Hardware and software aspects</td>
<td>62</td>
</tr>
<tr>
<td>3.2 Message communication</td>
<td>66</td>
</tr>
<tr>
<td>3.3 Performance measures</td>
<td>74</td>
</tr>
<tr>
<td>3.4 CUENET Applications</td>
<td>76</td>
</tr>
<tr>
<td>3.5 TOMP - Torino MultiProcessor</td>
<td>78</td>
</tr>
<tr>
<td>IV. Stochastic Petrinet (SPN) modeling of CUENET</td>
<td>82</td>
</tr>
<tr>
<td>4.1 Modeling assumptions</td>
<td>82</td>
</tr>
<tr>
<td>4.2 Estimation of System parameters</td>
<td>84</td>
</tr>
</tbody>
</table>
4.3 Two processor SPN model
4.4 Modified Two processor SPN model
4.5 Three Processor SPN model
4.6 Modified Three Processor SPN model
4.7 Analysis of SPN models
4.8 Discussion of SPN models of CUENET
V. Generalized Stochastic Petri Nets (GSPN)
       modeling of CUENET
      5.1 Preliminaries
      5.2 Two Processor GSPN models
      5.3 Three Processor GSPN model
      5.4 Analysis of GSPN models
      5.5 Discussion of GSPN models of CUENET
      5.6 Aggregated GSPN models for two Processor CUENET
      5.7 Discussion of Aggregated GSPN models of CUENET
VI. Conclusions
References
Appendix A
Appendix B
Appendix C
Appendix D
Appendix E

LIST OF FIGURES AND TABLES

FIGURES

1.1 THROUGHPUT - "Saturation effect" 
2.1 Example of a Petrinet
2.2 Example of a GSPN
2.3 Time behavior of GSPN
3.1 Configuration of CUENET
3.2 Message control mechanism in CUENET
3.3 State diagram of a CUENET Processor
3.4 Two Processor architecture of TOMP
4.1 Two Processor SPN model - SPN2A
4.2 Modified Two Processor SPN model - SPN2B
4.3 Three Processor SPN model - SPN3A
4.4 Modified Three Processor SPN model - SPN3B
5.1 Two processor GSPN model - GSPN2A
5.2 Modified Two Processor GSPN model - GSPN2B
5.3 Two processor (with message queues) GSPN model - GSPN2C
5.4 Three Processor GSPN model - GSPN3
5.5 Two processor Aggregated GSPN model - GSPNR2
5.6 Two processor (with message queues) Aggregated GSPN model - GSPNRQ2
5.7 Three Processor version of GSPNRQ

14
27
51
54
63
69
75
79
86
91
93
97
111
113
116
120
135
140
144
TABLES

4.1 Details of Model and Reachability set size for SPN models of CUENET 100

4.2 Computation times for SPN analysis 104

4.3 Comparision of CUENET and TOMP SPN models 104

4.4 SPN analysis results of SPN2A 106

4.5 SPN analysis results of SPN2B 107

5.1 Details of model and reachability set size of GSPN models of CUENET 126

5.2 Computation times for GSPN analysis of GSPN models 127

5.3 GSPN analysis results of GSPN2A 129

5.4 GSPN analysis results of GSPN2B 130

5.5 GSPN analysis results of GSPN2C 132

5.6 GSPN analysis results of GSPN3 133

5.7 Details of model and reachability set size for Aggregated GSPN models 146

5.8 Computation times for GSPN analysis of Aggregated GSPN models 147

5.9 GSPN analysis results of GSPNR2 149

5.10 GSPN analysis results of GSPNRQ2 150

(Processing Power for different queue sizes)

5.11 GSPN analysis results of GSPNRQ2 151

(C-Bus Throughput for different queue sizes)
CHAPTER I

INTRODUCTION

1.1 Evolution of Distributed computing systems:

The motivating factors for the continuing development of computing systems are[ENSLO-77]:

(1)speed of computation and throughput,

(2)Flexibility,

(3)Availability,

(4)Reliability.

Initially the computer system manufacturers concentrated on increasing the speed of computation and throughput by increasing the speed of the hardware circuitry and then by developing efficient operating system software [ENSLO-77,LIEBO-81]. Thus resulted the complex centralized main frame systems like the IBM system 360 and 370 series and recent 4300 series(4331,4341),DEC's series 10. As the applications of computers grew in the areas like banking, patient monitoring and industrial process control the other motivating factors(i.e. Flexibility, Availability and Reliability) along with increased computing throughput, gained importance. These system requirements caused the computing system designers to think in terms of multiple processing units interlinked in some fashion, thus giving rise to multiple processors systems and Distributed Computing Systems[LIEBO-81]. Enslow characterizes a fully Distributed Computing System(DCS) by the following
attributes [ENSLO-78]:

1. A multiplicity of general purpose components including both physical and logical resources, that can be assigned to specific tasks on a dynamic basis.

2. A physical distribution of these physical and logical components of the system interacting through a communication network.

3. A high level operating system that unifies and integrates the control of the distributed components. Individual processors can have their own operating systems.

4. Cooperative autonomy, characterizing the operation and interaction of both physical and logical resources.

5. System transparency, permitting service to be requested by name only.

Generally these characteristics are present in varying degree in the actual systems thus providing varying shades of distribution of control, computation and processor interaction. Thus there exists a wide diversity of system architecture, control and processor interaction in DCSs. There are various categories depending on the nature of location, degree of coupling, processor logical relationship, processor connectivity and so on. One such categorization based on the degree of coupling and processor connectivity classifies DCSs as:

(a) Multiprocessors,

(b) Computer networks,

(c) Local area networks.
(a) Multiprocessors:

A Multiprocessor is a DCS which has the following characteristics [ENSLO-77, SATYA-80]:

* contains two or more processors with approximately comparable capabilities,
* these processors are tightly coupled and communicate through shared memories,
* all processors share access to input/output channels, control units and secondary storage devices,
* the entire system is controlled by one operating system providing interaction between processors, their programs etc.

The concept of multiprocessors is not new and some early examples of them are IBM 360/65, Burroughs D8.5, Bendix G21, IBM 370/168 and CDC Cyber 170 [FULLER-78, SATYA-80, ENSLO-77]. In these early systems, the degree of tight coupling is quite high in that each processor accesses the common memory for each program code execution and data manipulation [WEITZ-80, FATHI-83]. With the advances in semiconductor memories and integrated circuit technology, the present day mini or microcomputer has its own private memory and also executive software. In such multiple computer systems, employing tight coupling, the shared memories are employed for moving data from one processing element to the other. The program execution is done in the private memory of the processors. Here the degree of tight coupling is moderate.
Such DCS are also categorized as Multiprocessors. Some recent examples of such multiprocessors are: Cm* developed at the Carnegie Mellon university [FULLER-78], Bell Canada's DATAPAC SL-10[WEITZ-80], Torino MultiProcessor(TOMP) developed, as part of the Municr project, at Polytechnico De Torino, Torino, Italy [CONTE-81].

(b) Computer networks:

Computer networks are DCSs possessing the following characteristics [TANEB-81]:

* Each processor is a fullfledged conventional computer systems with independent Input/Output devices and secondary memories.

* the processors are geographically dispersed and loosely coupled,

* inter processor communication is by passing messages in the form of packets over serial communication links under the control of communication controllers which route the message in an optimum fashion,

* each processor has its own operating system with an overall network operating system for controlling the intercomputer communication.

The computer networks are also known by the names, Long haul networks and public data networks. ARPANET, TYMNET are examples of computer networks.
(c) Local area networks (LANs):
The rapid and continuing advances in integrated semiconductor technology (LSI, VLSI) gave rise to the development of low cost microprocessors. Now microprocessors are available from 8 bits to 32 bits with varying computational power and their cost is still on the decreasing trend [FULLER-78, GUPTA-84]. With the advent of these low cost, high capability microprocessors, it is now possible to perform computations at the site of data. This has resulted in intelligent terminals, controllers, personal computers and various consumer goods [FULLER-78]. Also interest grew in interconnecting such diverse computing systems within an organization leading to the evolution of LANs.

A LAN has the following characteristics [TSAO-84, CLARK-78]:

* limited in geographic scope—within 0.1 to 1.0 km range, usually linking processors within a building or within a single floor,

* provides high bandwidth, short delay communication (over 1Mbits/sec) over inexpensive communication media (twisted pair wires, coaxial cables),

* processors are loosely coupled and communicate by passing massages over the communication medium of the network,

* employs simple routing and control algorithms as
opposed to the long haul computer networks. Examples of early Local area networks are ETHERNET, WANGNET etc (CLARK-78).

The typical environment where LANs are applicable today are [TSAO-84]:

1. Office automation- supporting functions like electronic mail, text processing, document distribution and voice storage.

2. Universities- to provide communication network to access central or distributed facilities, special purpose application software etc.

3. Factories- help in automating and controlling manufacturing techniques such as CAD/CAM, robotics, numerically controlled processes etc.

4. Hospitals- to provide communication support for patient monitoring, patient file retrieval etc.

Several different configurations and communications techniques are possible in the case of LANs, since they do not have the same operational restrictions of long haul computer networks.

Several educational institutions and laboratories have built and been building different types of multiple microprocessor DCS with tight coupling and loose coupling, for studying their performance and try different alternative architectures [MARSAN-83]. The Cm* at the Carnegie Mellon
University, X-tree at the University of California, Berkeley, MICRONET at the University of New York at Buffalo, [WITTIE-80], TOMP at Polytechnico De Torino, Torino, Italy [CONTE-81] and CUENET[GROSS-82] are some examples.

The CUENET(Concordia University Educational NETwork) is loosely coupled LAN comprising of 8 and 16 bit microprocessor systems, developed here at the Department of Computer Science, Concordia University, Montreal, Canada.

1.2 Factors in the design of DCS:
The major aspects of the design of a DCS and their applications lies in the design or selection of the following:

(a) A set of processors(homogeneous or heterogeneous) and memories that are interconnected.

(b) An interconnection network that physically connects the various processors and memories.

(c) Subdivision of the sequential program into subprograms that can be run concurrently on the various processing elements.

(d) An operating system that will coordinate the execution of different subprograms, allocate the subprograms to different processing elements and manage them.

There are various ways these design steps can be carried out and also there areas which present interesting and challenging problems for research. Some of the problems a
designer of a DCS has to deal with are [FULLER-78]:

1. Task decomposition: How should tasks executed on a uniprocessor be partitioned so that they can be run on a set of smaller processors? The task decomposition must take into account the architecture of the DCS on which it is intended to run. The optimum way this partitioning can be done is still an open problem [KUCK-77, GAJSK-85].

2. Interconnection structure: What are the most effective types of processor/memory and processor/processor interconnection structure? How flexible the interconnection structure be?

3. Operating system structure: What kind of operating system is suitable for controlling, resource managing, distribution of tasks among the processing elements and their protection in a distributed computing system comprising of hundreds of processing elements?

4. Interprocessor interference: The partitioned tasks distributed on the processing elements in a DCS will need to cooperate with each other so that they can provide the requisite service or perform the required computation. This would involve contention for memories, I/O devices for communication paths. These must be kept to the minimum.

5. Dead lock avoidance: With multiple processors contending for resources a dead lock situation can develop, where each of a group of processors is waiting for resources assigned to the other processors in the group, and none of the processors in the group being in a position to proceed until
its demands are satisfied. The designer must provide a facility for avoiding the dead lock or for taking remedial measures in the event of a dead lock.

6) Input-Output: How the Input/Output devices in general and secondary storage in particular are integrated into a DCS?

7) Fault tolerance: How one can utilize the hardware and software structure of a DCS to provide capability to survive failures of components in the system?

Different designers have overcome these problems differently. Thus each DCS performs in different ways.

1.3 Performance Evaluation of DCS:

The term "Performance" of a computer system has many different interpretations. Normally "Performance" answers questions like "How well does the system enables one to do what he wants to do?" and "How well the system does what it is intended to do?"

The Performance of a system can be discussed from two different viewpoints: first the effectiveness with which the system handles a specific application and second the internal efficiency of the system [SVOBOD-76].

Effectiveness is what is seen by the system user. Efficiency is how the system uses its resources in order to process the respective workload. System effectiveness, of course, is influenced by the internal efficiency.
Thus the performance of a DCS is the effectiveness with which the various processors, memories and other devices, the intercommunication structure are utilized under a particular workload.

The system components influence the system performance through their own characteristics and through their mutual interaction. The performance of a system is assessed with respect to a set of parameters which are termed Performance measures.

Performance Evaluation is the process of evaluating the performance of a system. The major steps involved in performance evaluation of a system are: (1) Definition of the relevant performance measures, (2) Determination of the values of the Performance measures with respect to the system structure and system workload.

Performance evaluation of a DCS is the study of how the various components, both hardware and software, interact with each other and how the flow of data and information is maintained and controlled in the interconnection structures and how efficiently the system functions [AGARWA-83]. The efficiency of a DCS depends upon many factors related to different ways hardware and/or software components interact i.e. contention for shared resource, synchronization of tasks etc. By doing a performance evaluation study of a system one can know how efficiently the various processing
elements are structured, the efficiency of interprocessor communication schemes and the distributed software [MARSAN-83].

We will discuss the various performance measures of DCS and different methods of performance evaluation.

1.3.1 Performance measures of DCS:

Performance measures are a set of quantitative parameters which characterize the performance of a system [SVOBOD-76, WEITZ-80].

The performance measures of a conventional computing system, principally are, the throughput, turnaround time, response time and system availability.

The throughput is the amount of work completed in unit time for a given workload. It is normally expressed as MIPS (Millions of Instructions Per Second).

The turnaround time is the elapsed time between submitting the job and receiving the output. Normally it is expressed in secs or minutes. Normally this is for batch jobs.

Response time is the turnaround time of requests and transactions in an interactive real time systems.

Availability of the system is the percentage of time a system is available to the user.
In a DCS the performance of the system depends on the individual capabilities of the processing elements, the type of interaction between them, the type of interconnection network and the way contention for a shared resource is managed. Here each processor will have individual performance measures similar to the conventional systems. Besides these, the main performance measures of DCS are processing power, throughput, waiting time for a shared resource like the global bus or the common memory.

The processing power for a distributed computing system expresses the amount of speedup or the increase in actual processing capability over a uniprocessor. Ideally, the processing power of a DCS with 'n' processors must be 'n' (assuming unit processing capability for each processor). In reality, this is not so since, the processing elements in a DCS spends some processing time in sending and/or receiving information from other processing elements. The processing power expresses the effective processing capability over a uniprocessor. Hence processing power P is expressed by the following expression:

\[ P = E(N_a) \] ..................................................(1.1)

where \( N_a \) = number of active processors in the DCS.

Processing Efficiency of a processor in a DCS is given by the following relation.

\[ \text{Processing Efficiency} = \left( \frac{P}{N} \right) \times 100 \] ...........................................(1.2)

where \( N \) = Number of processors in the DCS.
Throughput is a measure of the amount of work done per unit time. It is normally expressed as number of jobs or programs executed or messages transmitted per second. The unit for the throughput depends on whether one is looking at the system as a whole or at a subsystem. While one is considering the complete DCS, the throughput is proportional to the speedup or processing power P. When one is looking at the intercommunication subsystem then it is the number of messages transmitted. The throughput of a system or a subsystem can be given by the following expression:

\[ \text{Throughput} = \text{fraction of time the (sub)system is busy} \times \text{rate of operation} \]

(1.3)

Thus throughput expresses how effectively the system or the subsystem is functioning. Fig 1.1 shows the relationship between system throughput and the number of processors. Ideally the system throughput is to increase linearly with the number of processors (curve a). In practice the relationship would be more as shown by the curves b and c. In each case the system throughput levels off for a particular number of processors and then starts to decrease due to the "saturation effect" [FATHI-83]. The point at which the system throughput levels off and then saturates depends on a number of factors. Some of the factors are: the type of coupling, the nature of intercommunication network, the nature of application, the method of arbitration for the shared system resources etc. Thus the study of the throughput variation helps one in
estimating the optimum size of the system. The waiting time or queueing time for the various shared resources like global bus, common memory indicate how well the resources are utilized in a DCS.

These are some major performance measures which are considered in the performance modeling and measurement.

1.3.2 Methods of Performance Evaluation:
Performance evaluation of a system can be carried out by different methods. These methods are[AGARWA-83]:
(a) Performance Analysis,
(b) Simulation,
(c) Performance measurements.

(a) Performance Analysis:
Performance Analysis of a system is the study of how the various components of the system interact and affect the system's performance by representing the system and its components using a model [SVOBOD-76]. A model is an abstraction of the actual system representing the various system's components and their interactions. Also the model represents the main factors determining the system's performance in statistical or mathematical way [CHANDY-81]. From these relations the performance measures of the system can be computed. A wide spectrum of models can be derived depending upon the level of detail used for the representation of the system's behavior. A very detailed description of the system will lead to an extremely complex models where as a more abstract or higher level view of the system can result in a fairly simple model which is easier to analyse [MARSAN-83, BUZEN-77]. Thus different kinds of models are used in performance analysis of computer systems. Some of the types of models used are: queueing (network) models, stochastic and graph models like stochastic Petrinets, Generalized Stochastic Petrinets and Markov models [KLEIN-75, MARSAN-83]. These models possess markovian property which makes the derivation of the various performance measures tractable. For these models usually probabilistic workloads are assumed [SVOBOD-76].
Performance analysis is normally employed for a system which is in its design stage or early stage of realisation and for a system on which the measurement of various parameters is involved or is not cost effective [SVOBOD-76]. The performance analysis is also used to understand the interaction of the various components of a system and their effect on the system's performance so that with minimal measurement of certain parameters one can monitor the performance of the system.

(b) Simulation:
The technique of simulation can be viewed as a combination of modeling and measurement. The simulation process requires a model of the system and a simulator. The simulator is a mechanism that simulates the system behavior as specified by the functional model of the system and the work load. It yields the necessary data required for performance analysis [SVOBOD-76].

Simulation helps in estimating the performance of new designs and new configurations before their implementation. The complexity of simulation depends upon the level of detail included in the model. Simulation based on a very detailed model becomes very costly both in development and use. For performance analysis, a simulator must only simulate those events that change the system's state. Such simulators are called Discrete Event simulators. These simulators are quite popular for simulating computing
systems. The development of simulation languages like GASP (General Activity Simulation Program), GPSS (General Purpose System Simulator) and SIMSCRIPT provide the basic support for building discrete event simulator for a system under study.

(c) Performance Measurement:
Performance measurement constitutes part of performance evaluation where certain data like amount of CPU secs, number of messages received and/or sent by each processor, waiting time for each processor while contending for the shared resource etc are determined quantitatively. These measurements are useful in the determination of the various performance measures of the system under study. Normally some kind of monitoring scheme is employed for the measurement of these data. There are hardware monitors, software monitors and hybrid monitors [NUST-75]. The choice of a particular type of monitoring scheme is dependent upon what type of data to be collected, the amount of overhead that the system can tolerate, the location of the monitoring facility and so on [AMER-82]. In general a suitable hybrid monitoring scheme is employed.

Some practical examples of monitoring systems used for performance measurement are:
(a) The Integrated Instrumentation Environment (IIE), is a comprehensive performance monitoring scheme which emphasizes an integrated experiment management concept. It is
implemented on Cm* multiprocessor [SEGALL-83].

(b) A hybrid monitor consisting of a hardware monitor called ZAHLMONITOR III and a software monitor, designed for the instrumentation and performance monitoring of the Erlangen general Purpose Array (EGPA) [FROMM-83]. The EGPA is a hierarchical multiprocessor array.

The term performance evaluation refers to both the performance analysis and performance measurement which are complementary to each other [SVOBOD-76]. The analytical model provides the framework for the measurement and the measurement furnishes data for validating the analytical model. Further the analytical model aids in testing a hypothesis and finding solution to performance problems while the correctness of the predictions of the model is finally verified by measurement. Thus while doing the performance evaluation of a system both the performance analysis by modeling and the measurement of the performance measures derived must be carried out. Normally at the design and early implementation stage analysis by modeling is carried out. Once the system implementation is complete and running fairly well the measurements of the various performance measures must be done so that the analytical model is validated and also improved.

1.4 Thesis motivation and outline:

In the preceding sections we have discussed the evolution of DCS and main thrust behind the development of both tightly
coupled and loosely coupled DCS at various universities and laboratories.

It is observed that it is easier to model the communication aspects in a tightly coupled systems than in loosely coupled systems [REID-82]. The reason according to Reid, is the difficulty in obtaining the bounds on the number of messages and the number of resources required in a loosely coupled system. This seems to be corroborated by the fact that recently published performance analysis studies have been on tightly coupled systems. Some examples are: Performance analysis of Torino MultiProcessor(TOMP) by Marsan et al [MARSAN-83, MARSAN-84, MARSAN-85] and performance modeling of Fault Tolerant Multiprocessor (FTMP) by Woodbury et al [WOOD-84]. In these performance modeling the graphic models Stochastic Petrinets(SPNs) and the Generalized Stochastic Petrinets(GSPNs) have been used for easy system description. The existence of a loosely coupled DCS(CUENET) in our department and the recent interest in graph models motivated us to attempt this performance evaluation study using SPNs and GSPNs.

Chapter II discusses the modeling requirements of DCS, describes the graph models SPNs and GSPNs used in our study. Chapter III briefly discusses the hardware/software architecture, message communication, aspects and performance measures of CUENET. Chapter IV deals with the SPN models for 2 and 3 processors configuration of CUENET. Here the
steps involved in the analysis of the models and the limitations of SPN in modeling CUENET are discussed. The GSPN modeling of CUENET is dealt with in Chapter V. The last chapter deals with the conclusions and suggestions for future work.
CHAPTER II

MODELING TECHNIQUES FOR PERFORMANCE ANALYSIS OF DISTRIBUTED COMPUTING SYSTEMS

2.1 General requirements for modeling Distributed Computing Systems (DCS):

In general a DCS consists of a set of physically separate processors which are connected to each other by an interconnection network. The nature of the DCS varies depending on the extent of coupling among the various processors and the distribution of control of various functions of the system. The processors can be "tightly coupled" where they communicate amongst them by means of shared memories or they can be "loosely coupled" where they communicate by passing messages to each other over the interconnection network. The control of the system can be done from a central site or it can be decentralized so that each processor can function in an autonomous fashion [PATHI-83, WEITZ-80].

As discussed in chapter I, the performance of a DCS depends on a variety of factors:

1) The number and nature of processors,
2) The degree of coupling among them,
3) The type of control and the kind of interconnection
network which connects all the processors,
4) the type of applications that can be or is run on the system.

All these factors have varying degree of effect on the performance of the DCS. In order to study the effects of these factors on the system's performance, various modeling techniques have emerged over the years [MOLLOY-81, MARSAN-83].

To have meaningful study of the DCS being modeled, the modeling technique must have certain attributes [MOLLOY-81]. They are:

1) Logical correctness— the model must depict clearly and correctly the logical relationship of components of the system and be able to handle varying degree of logical complexity.

2) Flexibility— The modeling technique should allow for varying degree of abstraction. For instance, the abstraction may be at the level of software modules or at the level of instructions.

3) Ease of representation of concurrency— the modeling technique must allow easy representation of concurrency and conflicts within the system.

4) Deal with performance issues— the modeling technique must provide good facility to define certain performance parameters and determine their values mathematically.

Some of the modeling techniques employed in the performance analysis of DCSs are:
a) Stochastic models,
b) Graph models,
c) Integrated models.

a) Stochastic models: They include Queueing network models and direct Markov models.

In Queueing network models, the various components of the system are represented by servers to which jobs/customers arrive at a particular rate and the server provides the service at a particular rate [KLEIN-75, CHANDY-81]. Usually exponential distributions are assumed for the various arrival and service rates. In the queueing network, the movement of the jobs among the servers is represented by a discrete Markov chain.

In direct Markov chain models, the various states the system goes through is represented by a finite Markov chain. Here the way the various system states are defined is very important.

In the above two stochastic models the emphasis is on the performance issue.

b) Graph models: In these models the logical relationship between the system components can be easily represented due to the graphical nature of these models [MOLLOY-81]. For modeling interaction between software modules control graphs and precedence graphs have been employed. The standard Petri nets, originally proposed by C.A. Petri [PETER-81] enable easy representation of concurrency and conflict within a
DCS. In these graph models the logical correctness can be easily checked by applying certain basic rules in their construction [MOLLOY-81,PETER-81].

C)Integrated models: In these models effort is made to integrate both the logical correctness and performance issues. One such model is the Timed Petrinets[ZUBER-80] which associates a fixed time interval with each transition of a standard Petrinet. In timed petrinets, due to the deterministic time, modeling of conflicts within a system becomes difficult[MOLLOY-81].

In another method of integration, the integrated model is obtained by introducing stochastic properties within a graph model. This integration can be best viewed when the interface of separate correctness and performance modeling is considered. Graph modeling takes the structure as input and by formal rules constructs the state space of operation. Performance modeling starts with a state space and makes assumptions on the timing of operations (typically memoryless to make it markovian) and calculates delay and throughput measures in a straightforward manner. For a graph model which is stochastic these two operations overlap. This allows a testing procedure to be employed which investigates the resulting state space of the graph model for logical correctness while ignoring time. Then performance testing procedure (probably automated) can be applied on the verified state space.
The stochastic Petrinets [MOLLOY-81] proposed by M.K. Molloy and the Generalized Stochastic Petrinets [MARSAN-84] proposed by Marsan et al belong to this type of integrated models. These integrated models provide a good integration of logical and performance modeling power for studying the performance of DCS.
2.2 Petri nets:
2.2.1 General Description and Definitions:
Petri net is a graph model originally proposed by C.A. Petri and later developed by other researchers for modeling computer systems [PETER-81]. The Petri nets have recently gained more popularity for the modeling of DCSs, because of the ease with which they model concurrency, conflicts and synchronization between processors or processes.

Petri net, from graph theory point of view, can be thought of as a bipartite, directed graph comprising of objects called Places and Transitions [AGERWA-79].

Definition 2.2.1: The structure of a Petri net C is given by the triple,

\[ C = (P, T, A) \] ......................................................... (2.1)  

where, \( P = \{p_1, p_2, \ldots, p_n\} \) is a finite set of places,

\( T = \{t_1, t_2, \ldots, t_m\} \), is a finite set of transitions,

and \( n, m \geq 0 \).

\( A = (TXP)U(PXT) \) is a set of arcs.

The places and transitions together represent a set of events or actions being performed by the system being modeled. The way the places and transitions are connected by the directed arcs represent the structure of the system under study.

If there is a directed arc from a place to a transition, then the place is termed input place of that transition.
Similarly, if a directed arc exists between a transition and a place, then the place is termed as output place of that transition. In general a place can be connected to a transition by more than one directed arc.

Graphically, the places are represented by circles, transitions by bars and tokens by small dark dots within a circle. Fig 2.1 gives an example of a Petrinet.

![Figure 2.1 Example of a Petrinet](image)

This Petrinet has 5 places and 5 transitions. The set $P$ is given by \{ $p_1, p_2, p_3, p_4, p_5$ \} and the set $T$ is given by \{ $t_1, t_2, t_3, t_4, t_5$ \}.
Definition 2.2.2: A place is called an input place of a transition if there exist a directed arc from the place to the transition.

Definition 2.2.3: A place is an output place of a transition if there exist a directed arc from the transition to the place.

Definition 2.2.4: Input function \( I(t) \) of a transition is the set of places which form the input places to the transition.

Thus, \( I(t) = \{ p \mid (p, t) \in A \} \) \hspace{1cm} (2.2)

Example, In fig.2.1, \( I(t_1) = \{ p_1 \} \).

Definition 2.2.5: Output function \( O(t) \) of a transition \( t \) is the set of places which form the output places to that transition.

Thus \( O(t) = \{ p \mid (t, p) \in A \} \) \hspace{1cm} (2.3)

Example, in fig.2.1, for the transition \( t_1 \), \( O(t_1) = \{ p_2, p_3 \} \).

Each place in Petrinet can hold a set of tokens representing certain event or condition.

Definition 2.2.6: A marking \( M \) of a Petrinet is the assignment of tokens to the places in a Petrinet. \( M : P \rightarrow I \) where \( I \{ 0, 1, 2, \ldots \} \).

\( M \) can also be viewed as a vector whose \( i \)th component \( m_i \) represents the number of tokens assigned to the place \( p_i \).

Thus \( M \) is defined as,
\[ M = \{ m_1, m_2, \ldots, m_n \} \] \hspace{2cm} (2.4)

Example: In fig. 2.1 the petrinet has the following marking: \( (1, 0, 0, 0, 0) \). A petrinet \( C \) with marking \( M \) is a marked petrinet.

Definition 2.2.7: The structure of a marked petrinet \( PN \) is given by a four tuple,
\[ PN = [P, T, A, M] \] \hspace{2cm} (2.5)

Definition 2.2.8: A transition is "enabled" if each of its input places has at least as many tokens as the number of arcs connecting that place to the transition.

Definition 2.2.9: An enabled transition can "fire" in a Petrinet by removing all the enabling tokens from its input places and depositing in each of its output places one token for each arc connecting that place.

The firing of an enabled transition \( t \) in a given marking \( M_0 \) produces a new marking \( M_1 \) which are related by the following relation.
\[ M_1 = M_0 - I(t) + O(t) \] \hspace{2cm} (2.6)

Definition 2.2.10: A Petrinet is "executed" by firing all the enabled transitions in a given marking.

Thus in executing a marked Petrinet, the position and number of tokens change in the marked Petrinet. Also a set of markings will be produced by the firing of transitions during the execution of a marked Petrinet.

Definition 2.2.11: Reachability set \( R(PN) \) of a marked
Petrinet is the set of all markings which are reachable from a given marking $M_0$.

**Definition 2.2.12:** A marking $M_i$ is immediately reachable from a marking $M_0$ if and only if some enabled transition $t$ in the marking $M_0$ yields $M_i$.

**Definition 2.2.13:** A marking $M_i$ is reachable from marking $M_0$ if it is immediately reachable from marking $M_0$ or is reachable from any marking which is immediately reachable from $M_0$.

**Definition 2.2.14:** A transition $t$ in a marked Petrinet PN is "live", if for each $M_i \in R(PN)$ there exists a marking reachable from $M_i$ in which $t$ is enabled.

**Definition 2.2.15:** A marked Petrinet PN is "live" if all of its transitions are live.

**Definition 2.2.16:** A place in a marked Petrinet PN is $K$-bounded if and only if there exists a $k$ such that $m_i \leq k$ for all $M_i \in R(PN)$.

**Definition 2.2.17:** A marked Petrinet PN is $k$-bounded ($k$-safe), if for some fixed $k$ each place in it is $k$-bounded.

**Definition 2.2.18:** A place $p_1$ in a marked Petrinet PN is safe if it contains at most 1 token (i.e. 1-bounded).

Example: In fig2.1, for the given marking place $p_1$ is a safe place.
Definition 2.2.19: A marked Petrinet PN is safe if each place in it is safe.

In the Petrinet shown in fig 2.1 the place p₁ is having one token and the other places have zero tokens. The place p₁ is a safe place and it can be shown (see section 2.2.2) that the marked Petrinet is also safe, since no place will at any marking contain more than one token in it.

The liveliness and boundedness are very important properties of marked Petrinets which are useful in modeling systems for studying their operation and ascertaining their logical correctness [Murata-84].

Definition 2.2.20: A marking M₁ of a marked Petrinet PN is said to be a "terminal" marking if no transition in the Petrinet is enabled in that marking.

The presence of a terminal marking in the reachability set of the system modeled indicates that there are conditions under which the system can exhibit a deadlock. A marked Petrinet containing a terminal marking is not a live Petrinet.

2.2.2 Analysis of Petrinets:
The analysis of Petrinets mainly consists of determination of the various markings that are reachable from a given marking, which also form the reachability set of the Petrinet. Also during the analysis whether any "terminal"
marking(s) exist or not is determined. In general the reachability set of a marked petri net is finite. It is more so for a marked Petri net which is bounded and live.

The reachability set is determined by constructing a reachability tree which clearly depicts the relationship between markings and enabled transitions. In one marking more than one transition can be enabled. The resulting marking is found by using the relation (2.6). The successive generation of markings from the initial marking is followed till a known marking or a terminal marking is reached. The analysis is terminated when all enabled transitions and their resulting markings are examined.

The determination of reachability set of a marked petri net PN, in the similar lines as described above, can also be done by assigning unique and easily manipulatable number to each marking and also to each transition. For each transition is associated one number for its input places and one for its output places. There are two methods: one using prime numbers (as proposed by M.K. Molloy) [MOLLOY-81] and another method, devised by us, using binary and hexadecimal numbers. These methods are detailed below.

(a) Prime number method:

Here a distinct prime number is assigned to each place in the Petri net. A unique prime number called State number is associated with each marking of the marked petri net. Also
with each transition an input number and an output number (both prime numbers) are associated.

These prime numbers are determined in the following way: The marking of the marked Petri net describes the places which have tokens. The state number for this marking is determined by multiplying the various prime numbers associated with places having tokens. If the place has multiple tokens then the prime number is multiplied as many times as the number of tokens in that place.

Example: In fig 2.1, the given marking is \((1,0,0,0,0)\) and the prime numbers \(2,3,5,7,11\) are associated with the places \(P_1, P_2, P_3, P_4\) and \(P_5\) respectively. The state number for this marking is 2.

The input and output prime numbers for each transition is determined in the same manner, i.e. by multiplying the prime number associated with the input places and the output places.

Example: In fig 2.1, for the transition \(t_1\), input number is 2 (its input function consists of \(P_1\)) and the output number is \(3 \times 5 = 15\) ((\(P_2, P_3\)) form the output function).

(a) Determination of enabling of a transition in a given marking:
A transition is enabled in a given marking, if the state number of the marking is divisible by the input number of the transition.
Example: In fig 2.1, for the given marking \(1,0,0,0,0\) \(t_1\) is the only enabled transition. Thus the state number for this marking (2) is divisible by the input number of \(t_1\) (2).

(b) Determination of new marking:
The marking \(M_i\) immediately reachable from a given marking \(M_o\) by firing an enabled transition \(t_i\) is obtained by dividing the state number by the input number of \(t_i\) and multiplying by the output number of \(t_i\).
Thus, \(M_i = \frac{M_o \times O(t_i)}{I(t_i)}\).

Example: In fig 2.1, for \(t_1\) the new marking state number is \((2/2) \times 15 = 15\).

The steps for determining the markings in the reachability set in a given marked Petrinet, using this prime number method, are as shown below:
1) Describe the Petrinet in terms of input and output functions of each transition of the Petrinet.
2) Determine the input and output prime numbers for each transition as described earlier.
3) From the initial marking given, determine the initial state number.
4) Now for this marking determine which are the transitions that are enabled in this marking, as described in (a). If it is determined that no transition is enabled in the given marking, then that marking will be labeled as a "terminal" marking.
5) Determine the next marking, which is obtained by firing
an enabled transition, as shown in (b).

The number of tokens in each place in the given marking is determined by noting how many times the state number is divisible by the place prime number.

6) Check to see if this state number already exists if not give a new state number and increment the state count.

7) The steps 4 to 6 are repeated for each marking that are obtained.

8) The steps 4 to 7 are repeated till we get markings that are already existing or a terminal marking i.e. in step 4 no transition is enabled for that marking.

The program listing for the determination of the reachability set using the above steps is given in Appendix A.

Applying these rules for determining the reachability set for the given Petrinet shown in fig 2.1 we obtain its reachability set. There are 5 markings in its reachability set and they are as shown:

1 0 0 0 0
0 1 1 0 0
0 0 1 1 0
0 1 0 0 1
0 0 0 1 1

From this reachability set, we can infer that the given Petrinet in fig 2.1 is live and also for the given marking it is safe.
This prime number method has a drawback. As the number of places increases the size of the state number for a marking exceeds the word length (i.e. 48 bits in CDC Cyber 835), thus placing an upper limit on the number of places (with single token in each) it can handle. The upper limit is about 23 places.

To overcome this we devised the binary and the hexadecimal method.

(b) Binary and Hexadecimal method:

These methods adopt, in principle, the same steps in determining the various markings in the reachability set of a given marked Petrinet, as described in the previous method.

The major difference is in the way the state number, input and output numbers for the transitions of the Petrinet are determined. These methods overcome the place limitations of the prime number method.

In the binary method, a number to the base 2 is determined for each of the markings and the input and output functions of all transitions in the given Petrinet.

Example: The initial marking in the Petrinet shown in fig 2.1 is 1,0,0,0,0. The places \( p_1, p_2, p_3, p_4 \) and \( p_5 \) are assigned the weightage \( 2^0, 2^1, 2^2, 2^3 \) and \( 2^4 \) respectively. Thus the state number is given by \( 1 \times 2^0 + 0 \times 2^1 + 0 \times 2^2 + 0 \times 2^3 + 0 \times 2^4 = 1 \).

Similarly the binary number for the input and output
functions of each transition are determined.
Example: for the transition \( t_1 \) in fig. 2.1, the input function binary number is 1 and the output function binary number is \( 1 \times 2^1 + 1 \times 2^2 = 6 \).

Thus the various 1's in the state number for each marking indicate which of the places in the given Petrinet contain a token. Similarly the 1's in the input number of transition indicate the enabling conditions and the 1's in the output number of a transition indicate which places get a token when the enabled transition fires.

To determine which of the transitions are enabled in a given marking and the resulting new markings, we do logical operations on the state numbers, input and output numbers of transitions.

1. To determine whether a transition is enabled in a given marking or not, we do the following logical operations:
   (i) BitAND the state number and the input number of the transition.
   (ii) Check if the result is equal to the input number of the transition. If so the transition is enabled in the given marking.

2. To determine the new marking by the firing of the enabled marking we do:
   (i) \( \text{INTRES} = \text{State number bitANDed with (complement of the input number)} \).
   (ii) The new marking = INTRES BitORED with output number of
the transition.

Appendix B contains the program listing for this method.

In binary method, in one word, upto 48 places can be accommodated. The drawback is multiple tokens cannot be represented by this method. So we adopted the hexadecimal method where for each place we allot 4 bits in the state number thus providing upto 15 tokens at each place.

Here we determine the state number to the base 16.

Example: In fig 2.1 for the places \( p_1, p_2, p_3, p_4 \) and \( p_5 \) we assign weightage \( 16^0, 16^1, 16^2, 16^3 \) and \( 16^4 \) respectively. For the given initial marking shown in fig 2.1, the state number is given by \( 1 \times 16^0 + 0 \times 16^1 + 0 \times 16^2 + 0 \times 16^3 + 0 \times 16^4 = 1 \). The state number, in general, \( S.N. = n_1 \times 16^0 + n_2 \times 16^1 + \ldots + n_i \times 16^{i-1} \ldots + n_n \times 16^{n-1} \) Where \( n_i \) = number of tokens in place \( p_i \).

So for the given marking shown in fig 2.1, the state number is 1.

The program listing for this method is given in Appendix C. Here one word can handle upto 12 places and this program can handle upto 36 places.

The determination of enabled transition for a given marking and the resulting new marking is essentially on the same lines as done in binary case. The steps followed are:

1) We find the marking in terms of no of tokens in each place from the state number using the subroutine FNDDMRK.

2) The function ITRENA determines which of the transitions are enabled in a given marking.
3) The resulting marking obtained by the firing of an enabled marking is determined by the subroutine FNDNWST.

2.2.3 Applications:

Petri nets have been employed in modeling logic operations [HURA-81], proving correctness of concurrent processes [LAUTE-74] and modeling the logical behavior of computer systems [MARSAN-83]. Since Petri nets do not involve any timing considerations, it is difficult to model the various dynamic and time dependent aspects of computing systems for their evaluation. Various researchers have introduced the concept of time to marked Petri nets.

Some have introduced a fixed time period within which an enabled transition must fire. This is the basis for Timed Petri nets introduced by Zuberek [ZUBER-80]. With the use of Timed petri nets, modeling of conflicts in a system is difficult because of the deterministic time interval. Therefore Molloy [MOLLOY-81] and Shapiro [SHAPI-79] introduced probabilistic random firing of the various transitions of a Petri net.
2.3 Stochastic Petrinets (SPNs):

2.3.1 Preliminaries:

Stochastic petrinets were initially proposed by M.K.Molloy[MOLLOY-81]. A SPN, like a Petrinet, is associated with a finite set of places \( P \), a finite set of transitions \( T \), the set of input and output arcs \( A \) and an initial marking \( M \). In addition, a firing rate \( g_i \) is associated with each transition.

Definition 2.3.1: A Stochastic Petrinet structure is defined by the 5-tuple,

\[
SPN = [P,T,A,M,G] \tag{2.7}
\]

where \( P, T, A \) are as defined in (2.1) and \( M \) as defined in (2.4) and \( G \) is the set of firing rates for the transitions of the SPN.

\[
G = \{ g_1, g_2, \ldots, g_m \} \tag{2.8}
\]

The Petrinet shown in fig2.1 will become a SPN once we associate the set of firing rates \( G \) to it. In this example \( G \) is a vector of 5 elements, each element corresponds to a transition and specifies the rate at which that particular transition will fire.

2.3.2 Properties of SPN:

The SPN has all the properties of the underlying Petrinet, (as discussed in section 2.2.1), like finite reachability set, liveness and boundedness.
Further SPN has certain other properties by virtue of
the introduction of the probabilistic rates for the firing
of each transition in the SPN.

By associating a continuous distribution with each \( g_i \),
say an exponential distribution with its own mean, one
obtains a continuous time SPN. Similarly by associating,
with each of the firing rate \( g_i \), a discrete distribution
like geometric distribution with its own mean, one obtains a
discrete time-SPN.

The reasons for choosing the exponential or geometric
distributions for the firing of the transitions in a SPN are
discussed below.

A stochastic Process [KLEIN-75, MOLLOY-81] is a
collection of random variables \( X(t) \) defined on a common
probability space where \( t \) is the time variable and is the
subset of the time range (-\( \infty \),+\( \infty \)). Given a set of states \( S \)
and a stochastic process on the states \( S \), the process has
Markovian property if and only if the probability of
occurrence of a particular state as the next state depends
solely on the present state. Thus for any sequence of
states \( x_1, x_2, \ldots, x_n, x_{n+1} \) in the process,
\[
P(X_{n+1} = x_{n+1} | X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n)
= P(X_{n+1} = x_{n+1} | X_n = x_n)
\]  
(2.9)
The only probabilistic distributions that have this
"memoryless" property are the exponential and geometric
Molloy has shown that a finite place, finite transition, marked SPN is isomorphic to a one dimensional discrete Markov chain. Also he has proved that there is one to one mapping between the markings of the SPN and the states of the Markov chain. The rate at which the transitions take place between the states of the equivalent Markov chain is equal to the sum of the rates of the enabled transitions which would give a marking equivalent to the next state. There are certain properties of SPN which are useful in the analysis of the system being modeled.

2.3.3 SPN Analysis:
In SPN analysis, as in Markov analysis, ergodic (recurrent, non-null, aperiodic, irreducible) systems are of interest[HOEL-72]. They have steady state probability distribution of tokens which can be determined after the equivalent Markov chain is obtained. To assure the ergodicity of the underlying Markov chain of the SPN, some of the properties of the Petrinets are used. Molloy has shown that in a live, bounded Petrinet with an initial marking \( M_0 \) and a reachability set \( S \), there exists a unique reachability set of recurrent markings \( S' \), possibly a subset of \( S \), which entirely describes the steady state behavior of the Petrinet. Also any live, bounded continuous time SPN is ergodic or reducible to a single ergodic sub-chain and therefore has a solution for the steady state probabilities.
for the tokens in each place of the SPN[MOLLOY-81].

Thus the SPN, which combines the modeling power of the Petri nets and the mathematical richness of discrete Markov chain, is a good tool for studying the performance of DCSSs.

In SPN the dynamic behavior can be easily studied by constructing the reachability set of the SPN, by adopting the method for constructing the reachability set shown in section 2.2.2. In the process of determining the reachability set of the markings of the system, the basic properties of the associated Petri net like liveness, boundedness are also determined.

The various markings in the reachability set of the SPN form the different states of the isomorphic continuous Markov chain. In the SPN, as mentioned earlier, the various transitions are associated with a mean exponential firing rate. Hence the sojourn time in each marking (state) is an exponentially distributed random variable with average,

$$\sum_{i \in K} (r_i)^{-1} \tag{2.10}$$

where K is the set of transitions that are enabled by the marking. The transition rate from marking $M_i$ to marking $M_j$ is obtained as,

$$\sum_{l \in K_{ij}} r_l \tag{2.11}$$

where $k_{ij}$ is the set of transitions (which may contain only
one element in most of the cases) enabled by the marking \( M_i \) and whose firing generates the marking \( M_j \). Thus the infinitesimal generator (or rate matrix) \( Q \) of the Markov chain can be obtained and the steady state probability distribution \( Y \) of the various states of the Markov chain can be determined by solving the equation,

\[ YQ = 0 \text{ or } QTY^T = 0 \]  \hfill (2.12)

Since the system is irreducible, the rank of \( Q \) is \((n-1)\) and thus we have \((n-1)\) independent equations in (2.12) \cite{Molloy81,RAU81}. To make all the \( n \) equations independent so that (2.12) can be solved to obtain the steady state probabilities, we introduce as the total probability constraint i.e. \( \sum_{i \in N} y_i = 1 \), in place of the last dependent equation in (2.12). This makes the (2.12) to have the form \( AX=b \) which is solved easily, to obtain the steady state probabilities \( y_i \).

After the determination of the steady state probabilities of each marking in the underlying Markov chain, the token densities at various places of the SPN are calculated. The token density \( \delta_i \) of a place \( p_i \) in a SPN is determined by the following relation.

\[ \delta_i = \sum_{j \in R(PN)} y_j m_{ij} \]  \hfill (2.13)

Where \( y_j \) is the steady state probability of the marking \( j \) and \( m_{ij} \) is the number of tokens in place \( p_i \) in the marking \( j \). The token densities of all places in a SPN are
determined using this relation (2.13).

2.3.4 Applications:
SPNs are suitable for modeling the various hardware/software aspects like contention for the shared resources (buses, memories etc), synchronization of tasks that influence the performance of a DCS [MARSAN-83].

The choice of firing rates of the various transitions depend on the workload assumptions and details of operations included in the model. For example, in a SPN modeling memory contention, the transitions modeling the request and access of the common memory fire as per the workload assumed. Other transitions, which represent the bus arbitration or release etc, whose effect is not that relevant in modeling the memory contention can fire at a very fast rate thus minimizing their effect on the performance measures of the SPN model.

Thus the countability of the markings and the memoryless property of the exponential (and geometric) distributions are the key factors which make the SPN amenable for modeling systems which exhibit concurrency and contention[MOLLOY-81].

The introduction of SPNs have established a link between two important classes of computer system models: the graph models and the probabilistic models[MARSAN-85a]. SPN, because of its graphical nature, provides an easy to use high level interface for the specifications of a model of
the system under study. Queueing theory had long been used for this purpose. The SPN has certain advantages over the queueing models [MARSAN-84]. First, they allow easy portrayal of concurrency and synchronization within the system, which are not easily described with queueing network models. Secondly, the SPNs allow the description of a system at different levels of abstraction, so that the user can choose the one that best suits his needs.

Also in the direct Markov models, the determination of the various states of the system is very difficult and requires good working knowledge of the mathematics of Markov chains. The analyst using SPN can generate a proper and complete model of the system by virtue of the structure of the SPN.

Thus the SPNs can be easily used by people who are not very familiar with the probabilistic modeling approach. Further the whole procedure of determining the states in the Markov chain underlying the SPN and the solution to the steady state probabilities of the various states can be automated.

But SPN has one drawback. Even with the small increase in the complexity or detail in the modeled system, the resulting Markov chain grows at a faster rate in its complexity. As a result, the numerical solution of the Markov states presents major computational problems.
As a starter, SPNs are ideal for modeling the system at a fairly high level of abstraction and to ascertain the logical correctness of the model. In order to model the system with finer details and at the same time maintain the solution complexity of the underlying Markov chain at a reasonable level, a derivative of SPN called Generalized Stochastic Petri nets (GSPN) has been proposed by Marsan et al [MARSAN-84].
2.4 Generalized Stochastic Petri Nets (GSPNs)

2.4.1 Definitions:

In SPNs we associate a random firing time or rate with each transition. Often this is not desirable since one would want to associate time with those events which are believed to have the largest impact on the system performance. This is more so in a system comprising of activities differing in orders of magnitude. Then it is conceivable to model the brief activities only from a logical point of view, and associate time with the longer lasting ones. This choice becomes particularly convenient if by doing so the number of states of the associated Markov chain can also be reduced, thereby reducing the solution complexity. This is the main thrust behind the GSPN proposed by Marsan et al [MARSAN-84].

GSPNs are derived from the SPNs by partitioning the set of various transitions into two disjoint subsets comprising of immediate and timed transitions.

Definition 2.4.1: Immediate transitions fire in zero time once they are enabled.

Definition 2.4.2: Timed transitions fire after a random (exponentially distributed) firing time.

A GSPN has the same graphical structure as SPN and in addition the timed transitions are represented by thick bars and the immediate transitions by thin bars. The firing rates are obviously associated with the timed transitions.
and they may also depend on the GSPN marking.

Definition 2.4.3: The Generalized Stochastic Petri net (GSPN) is defined by the five-tuple,

\[ \text{GSPN} = \{ P, T, A, M, R \} \] .................................(2.14)

where \( P, T, A \) are as defined in (2.1) and \( M \) as in (2.4). \( R \) is exactly similar to \( G \) in (2.8) but here \( m = \) number of timed transitions in the GSPN.

As in SPN, in GSPN also several transitions may be simultaneously enabled by a marking. Let \( H \) be the set of enabled transitions in a given marking. If the set of enabled transitions \( H \) comprises only timed transitions, then the enabled timed transition \( t_i(i \in H) \) fires with a probability \( r_i/R \) ................................. (2.15)

where \( R = \sum_{K \in H} r_K \) exactly as in SPN.

If \( H \) comprises both immediate and timed transitions, then only immediate transitions can fire. If \( H \) comprises zero or more timed transitions and only one immediate transition, then this immediate transition is the one which fires. When \( H \) comprises of several immediate transitions, then it is necessary to specify a probability density function according to which the firing transition is selected. The subset of \( H \) comprising all enabled immediate transitions together with the associated probability distribution is called a "Random switch". The associated probability function is called a "switching" distribution. Different
markings of the GSPN may originate a single random switch whenever they enable the same set of immediate transitions, upon which a single (possibly marking dependent) switching distribution may be defined. If the probability associated with an enabled immediate transition is zero, the transition cannot fire and thus it behaves as if it were not enabled. This condition, in some cases can be modeled in a simpler way by means of "inhibitor arcs". An inhibitor arc connects an input place to a transition and is represented by a line terminating with a circle rather than with an arrowhead at the transition. Thus the definition for an enabled transition (Definition 2.2.6) can be generalized to include the inhibitor arcs.

Definition 2.4.4: A transition in a marked Petri net is enabled if all of its "normal" places contain at least as many tokens as the number of arcs connecting them to the transition and all the "inhibitor" input places contain no tokens.

Thus the inhibitor arcs help in testing for zero token in a particular place. The inclusion of inhibitor arcs, in certain cases, simplifies the GSPN by reducing the number of random switches to be defined.

2.4.2 An example of GSPN:

Fig 2.2 shows a GSPN comprising of seven places and seven transitions of which three transitions $t_1, t_4$ and $t_5$ are
Switching probabilities:

\[
\begin{align*}
\Pr(t_2) & = \frac{m_3}{m_3 + m_4}, \\
\Pr(t_3) & = \frac{m_4}{m_3 + m_4}, \\
\Pr(t_6) & = \frac{m_4}{m_3 + m_4} \\
& \quad \text{if } m_3 \neq 0 \text{ and } m_4 \neq 0, \\
\Pr(t_7) & = \frac{m_3}{m_3 + m_4}, \\
\Pr(t_6) & = \Pr(t_7) = \frac{1}{2} \text{ if } m_3 = m_4 = 0.
\end{align*}
\]

Figure 2.2 Example of a GSPN.
timed and the remaining four transitions $t_2, t_3, t_6$ and $t_7$ are immediate transitions. Transition $t_1$ fires at a marking dependent rate equal to $u$ times the number of tokens in place $p_1$. Transitions $t_4$ and $t_5$ fire at fixed rates $v$ and $z$ respectively. Transitions $t_6$ and $t_7$ are two conflicting immediate transitions, they are always enabled simultaneously so that it is necessary to define a switching distribution for each marking where $m_7$ is larger than zero. The immediate transitions $t_2$ and $t_3$ may be simultaneously enabled if $p_3$ and $p_4$ contain tokens. Here a switching distribution must be defined for each marking in which $m_2, m_3$ and $m_4$ are greater than zero. Thus two random switches are to be defined for this GSPN. One possible switching distribution definition is shown in fig 2.2.

For the initial marking shown for the GSPN in fig 2.2 only the timed transition $t_1$ is enabled. After an exponentially distributed random time with an average $1/2u$ transition $t_1$ fires and one of the two tokens from $p_1$ is put in $p_2$. Now the immediate transitions $t_2$ and $t_3$ are enabled and any one can be selected to fire as per the switching distribution defined in fig 2.2. Here they are assigned equal probabilities. Assume that $t_3$ fires removing tokens from $p_2$ and $p_4$ and placing one in place $p_6$. Now two timed transitions $t_1$ and $t_5$ are enabled, transition $t_1$ fires with a probability $u/(u+z)$, where as $t_5$ fires with the probability $z/(u+z)$. Assume that $t_1$ fires first, one token moves from $p_1$ to $p_2$, thus enabling transition $t_2$. This transition
fires immediately, being the only enabled immediate transition, thus moving one token each from places $P_2$ and $P_3$ and putting one token in place $P_5$. The resulting GSPN marking is such that the timed transitions $t_4$ and $t_5$ are enabled, each of which can fire first with the following probabilities:

$$\Pr(t_4) = \frac{v}{v+z}, \quad \Pr(t_5) = \frac{z}{v+z}.$$  

Assume $t_4$ fires first, so that a token is moved from $P_5$ to $P_7$ and a token is put in place $P_1$. The two immediate transitions $t_6$ and $t_7$ are now simultaneously enabled and as specified by the switching distribution in Fig. 2.2, each can fire with probability $1/2$, so that the token in place $P_7$ can move either to $P_3$ or $P_4$. Now the timed transitions $t_1$ and $t_5$ are enabled and this way the Petri net execution continues.

In general, the reachability set of a GSPN is a subset of the reachability set of the associated Petri net because the precedence rules introduced with the immediate transitions do not allow some states to be reached. The reachability set of a SPN is, instead, the same as for the associated Petri net. The GSPN of Fig. 2.2 comprises of 17 markings, where as the reachability set of the associated Petri net comprises of 33 markings. Further, the reachability set of the GSPN can be divided into two disjoint subsets, one of which comprises markings that enable timed transitions only, while the other comprises
markings that enable only immediate transitions.

The crucial part in the definition of GSPN of a system under modeling, is in the definition of random switches for the immediate transitions. Usually this requires some ingenuity and good insight into the functioning of the underlying system.

2.4.3 Evaluation of the GSPN markings steady state probability distribution.
If we observe the various state changes in a GSPN with respect to time, we may find "multiple discontinuities" in the system state changes as shown in fig 2.3.

![Figure 2.3 Time Behavior of GSPN](image)
This is because of the sequential firing of one or more immediate transitions which are enabled in a particular marking. It should be noted that the stochastic process underlying the GSPN spends a nonnegative amount of time in those markings which enable timed transitions only; but it transits in zero time through markings which enable immediate transitions. This time behaviour of the GSPN is equivalent to the time behaviour of a Stochastic Point Process \( \{ X(t), t \geq 0 \} \) with a finite state space. A one to one correspondence exists between the GSPN markings and the SPP state space [MARSAN-84].

The states or markings in which the GSPN spends non-negative amount of time are called "Tangible" markings and those markings in which it spends zero time are termed "Vanishing" markings.

Similar to the SPN, a GSPN with a finite set of places and finite set of transitions (including both the timed and immediate transitions) is isomorphic to a one-dimensional Markov chain which in this case is an "Embedded Markov Chain" (EMC). The following properties and assumptions regarding the GSPN will help in fully categorizing the associated EMC [MARSAN-84]:

1) The GSPN has a finite reachability set which is usually a subset of the reachability set of the underlying Petri net. Also there will not be any terminal (or absorbing) state in the reachability set of the GSPN due to the liveness
property of the underlying Petrinet. The finiteness of the reachability set is ensured by the boundedness of the underlying Petrinet.

2) The initial marking is reachable with a nonzero probability from any marking in the reachability set. This is made possible by the liveness of the underlying Petrinet.

3) The firing rates for the timed transitions are not a function of time but are constants.

These points help further to specify the nature of the SPP, as finite, state spaced, stationary (homogeneous), irreducible, continuous time Stochastic Point Process. The associated EMC will be a finite state space, continuous time, irreducible stationary chain. In other words, the GSPN is isomorphic to an ergodic EMC whose states are equivalent to the various markings of the reachability set of the GSPN. This ergodic nature of the EMC ensures the existence of the limiting steady state probabilities of various states in the EMC and the calculations of steady state probabilities of tokens at each place of GSPN.

Let S denote the state space of the EMC and within S we can distinguish the Tangible and Vanishing states. This also helps in partitioning the state space and transition matrix. Let $K_S$ denote the number of states in the EMC state space and $K_T$ and $K_V$ indicate the number of Tangible and Vanishing states such that

$$K_S = K_T + K_V.$$
The transition probability matrix $U$ of the EMC can be written as follows:

$$U = A + B = \begin{bmatrix} C & D \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & E \end{bmatrix} \quad \cdots \cdots \cdots \cdots (2.16)$$

where $C$ has dimension $K_v \times K_v$, $D$ has dimension $K_v \times K_t$, $E$ has dimension $K_t \times K_v$ and $F$ has dimension $K_t \times K_t$.

The elements of $A$ can be obtained using the characteristics of the random switches and the elements of matrix $B$ can be obtained using the firing rates of the timed transitions as given by the relation (2.15).

The solution of the system of linear equations $Y = XU \cdot (2.17)$ yields the steady state probability distribution of the states of the EMC[MARSAN-84]. The vector $Y$ gives these steady state probability distribution of these states of the EMC.

This system of linear equations can be solved in the same manner as is done in SPN analysis; by assuming that the system remains in the vanishing state for a infinitesimally small time instead of zero, i.e. we can associate a very high firing rate 'x' for the firing of the immediate transitions and obtain the transitions rate matrix. After we obtain the steady state probability vector, the steady state probability vector for the original GSPN is obtained by taking limits for 'x' going to infinity. This method of solution has the computational complexity same as the SPN
and since the GSPNs involve large state spaces, the computational complexity is rather large. Further this method does not make use of the state bifurcation into Tangible and Vanishing states.

Hence Marsan et al. (MARSAN-84) have suggested a solution method which is computationally more efficient and also makes use of the tangible and vanishing state classification of the states of GSPN. This solution method is briefly described below:

Here a "Reduced Embedded Markov Chain" (REMC) is defined over the tangible states only. In order to do this it is required to compute the total transition probabilities among tangible states only. A tangible state may be reached from another tangible state either directly or through some vanishing states. These various possible ways must be reflected in the transition probabilities of the REMC. This is achieved in the following manner.

Let i, j represent arbitrary tangible states, while r and s represent arbitrary vanishing states. C_{rs}, D_{rj}, E_{is} and F_{ij} represent the elements in the submatrices C, D, E and F respectively of the transition matrix U of the original EMC. The total transition probability between any two tangible states i and j can be computed in the following way:

\[ U'_{ij} = f_{ij} + \sum_{\text{rev}} c_{ir} \text{Pr}(r \rightarrow j) \] (2.18)
where $\Pr(r\rightarrow j)$ represents the probability that the EMC moves from vanishing state $r$ to the tangible state $j$ in an arbitrary number steps, following a path through vanishing states only. Thus, the total transition probability matrix for the REMC can be obtained and it is denoted by,

$$U' = F + EG^\infty$$

(2.19)

where $G^\infty$ is the explicit expression for $\Pr(r\rightarrow j)$ for the various vanishing states in the original EMC and original transition matrix $U$. In order to determine $G^\infty$, we focus our attention on the matrix $A$ whose elements represent the transition probabilities of movement between Vanishing states and Vanishing state to Tangible states. The $k$th power of $A$ represents the probability of moving from any vanishing state $r$ to any other state of the original EMC in exactly $K$ steps.

Hence $A^K = \begin{bmatrix} C^K & C^{K-1}D \\ 0 & 0 \end{bmatrix}$

(2.20)

From this we have $G^K = \sum_{h=0}^{K-1} C^hD$

(2.21)

representing the probability of reaching any tangible state in moving from any vanishing state in no more than $K$ steps, visiting intermediate vanishing states only. When $K$ can vary to $\infty$ we have,

$$G^\infty = \sum_{h=0}^{\infty} C^hD$$

(2.22)

The irreducibility property of the EMC ensures that the spectral radius of the submatrix $C$ is smaller than
Hence \( \lim_{K \to \infty} C^K = 0 \) \hspace{1cm} (2.23)

With this result we can state that a limit for the sum,

\[
\lim_{K \to \infty} \sum_{h=0}^{K} c^h \text{ exists and is finite.}
\]

Also the submatrix \( C \) can be written as an upper triangular matrix by suitable reordering of the states of the EMC, so that there exists a value \( K_0 \leq K_v \), such that \( C^K = 0 \) for any \( K \geq K_0 \).

From these results the infinite sum for \( G \) reduces to a sum of finite terms and it is expressed as,

\[
G^\infty = \left( \sum_{h=0}^{K_0} c^h \right) D \hspace{1cm} (2.24)
\]

With the determination of \( G^\infty, U' \) can be computed.

The solution of the linear equations \( Y' = Y'U' \) \hspace{1cm} (2.25)
yields the steady state probability distribution \( Y' \) for the REMC.

The advantage of this solution method is two fold. First, the time and space complexity of the solution is reduced, since instead of solving a system of \( K_S \) linear equations we must now (in the worst case) find the inverse of \( K_vXK_v \) matrix and then solve a system of equation of \( K_t \) linear equations. It is known that the complexity of the Gaussian elimination solution is \( O(K^3) \). Hence the solution approach proposed by Marsan et al reduces the complexity of the solution of the GSPN from \( O(K_S^3) \) to \( O(K_t^3) + O(K_v^3) \).
The other advantage is by decreasing the impact of the size of the set of vanishing states on the complexity of the solution method, a greater freedom is available in the explicit specification of the logical conditions of the original GSPN. This enables larger systems to be conveniently modelled with greater detail using GSPN.
CHAPTER III

THE ARCHITECTURE OF CUENET

3.1 Hardware and Software aspects.
Concordia University Educational NETwork (CUENET) is a
reconfigurable network of loosely coupled multiple micro
computers which are interconnected by one or more buses
called C-Buses [GROSS-82]. Its network configuration is as
shown in fig 3.1. Each C-Bus is controlled by a C-Bus
controller which is a processor dedicated for controlling
the transfer of messages over the C-Bus. Each processor in
the network connects to a C-Bus through an interface called
C-Bus Interface. If a processor is connected to more than
one C-Bus then it will have more than one C-Bus interface,
one for each C-Bus to which it is connected.

CUENET comprises of a master computer, several slave
computers and network memory units (NMUs). The master
computer is responsible for the coordination of all the
computers in CUENET and also acts as the interface between
the network and the end user. The computational tasks
required by the end users are carried out by the slave
computers. The slaves will accept and perform the commands
of the master such as load a user program, obtain data/user
algorithm from some other location in the network, and start
or terminate a user routine. The network memory unit is
accessible to all the computers of CUENET, as a common data
Figure 3.1 Configuration of CUENET

I : C-Bus Interface
NMU : Network Memory Unit
Cntlr : C-Bus Controller
bank, through the C-Bus.

The C-Bus is a bit parallel, byte serial passive bus over which the CUENET processors communicate by passing messages under the control of the C-Bus controller. The C-Bus controller is responsible for resolving the contention for the use of C-Bus and ensuring the safe transfer of message to the receiver. In the C-Bus controller some special control function like incrementing the bus address lines, parity checking are implemented using special purpose hardware so that the C-Bus controller can be implemented using a general purpose microcomputer. Also this simplifies the Bus control software and aids in quicker transfer of message.

In CUENET the various slave computers need not be homogeneous. Infact a slave processor can be a special purpose processor such as a processor for signal processing etc. This heterogeneity allows that all processing elements in the CUENET need not be of the same power and capacity. It is possible to select the type and configuration of the processing elements so that they match as closely as possible to the user application requirements.

Further the master processor does not contain any special purpose hardware and thus any computer in the network can function as the master as long as its internal configuration is capable of running the master processor
software. This will enable quick restart of the network in the event of a failure of the master computer. In CUENET the master computer differs from the slave processors in having certain special rights. One such is loading the user's programs into slave processors and the other is setting up the configuration of the network as required by the user. In CUENET the reconfiguration of the network is variable under program control. This is made possible by providing an access vector at each processor which indicates what are the processors this processor can communicate with. Except the master processor the slave processors can only read this access vector but not alter it. Master processor can alter the contents of the access vector by sending a special message over the C-Bus thus reconfiguring the network.

Besides this each processor in the network has an interface lock control unit which prevents any accidental or unauthorized access to the C-Bus interface area. In the C-Bus interface there are two registers called Lock and Key registers. The contents of the Lock register is set to a predetermined value (combination). Access to the interface is allowed only when the communication software writes the exact combination into the Key register. Only system programs but not the application programs are supposed to know the "password" (key) combination. This hardware lock mechanism increases the reliability of the application
software by preventing any unintentional erroneous messages to other processors in the event of a program crash or malfunction. Further this lock mechanism allows general purpose microcomputers to be employed as processors in CUENET.

Thus in CUENET the system reliability requirements are amply met by the provision of multiple C-Buses and special purpose hardware for access control and message transfer and parity checking, at the same time capitalizing on the advantages of centralized control, viz, hardware and software simplicity and system flexibility to use general purpose microcomputers.

Presently the CUENET implementation comprises of a single C-Bus on to which three slave MC 6809 processors and one MC 68000 master processor are connected. The C-Bus controller is implemented using another MC 6809 processor. The C-Bus, in the present implementation, has length of about 100 feet, runs around our laboratory connecting the three 6809 slaves and 68000 master. It comprises 32 address lines, 8 control lines and 9 data/parity lines-a total of 49 signal lines (twisted pairs).

3.2 Message Communication:
In the CUENET the various processors interact with each other by passing messages over the C-Bus under the control of the C-Bus controller. The C-Bus interface of each
processor has two communication buffers— an output buffer and an input buffer. These buffers are accessible by the individual processors and the C-Bus controller. The output buffer in the C-Bus interface holds a message which is destined to another processor. The input buffer holds a message which is sent to this processor by the other processors of the network. Apart from this, each processor has resident communication software which is executed by the processor for either sending or receiving a message. The special purpose hardware in each of the C-Bus interfaces and message communication software in conjunction with the hardware and software of C-Bus controller perform the message communication in CUENET. A message communication scenario in CUENET is described below:

The sender processor puts the message it wants to send in the output buffer of its C-Bus interface and draws the attention of the C-Bus controller. The C-Bus controller examines the output buffer of the sender to determine the receiver address and finds if the input buffer of the specified receiver is free. If it is free, the C-Bus controller deposits the message into the receiver's input buffer and marks the output buffer of the sender empty and the input buffer of the receiver full. In CUENET because of the loose coupling of the processors, the sending processor can be active working on the application program in its memory while the C-Bus controller is transmitting the message. Likewise, the receive processor can be operating on
its program till it gets the signal that it has received a message. This type of message communication enhances the parallelism of the distributed processing system.

In CUENET the messages sent by various processors during the execution of the user’s application programs follows a fixed format. The message format consists of a header, body of the message and footer. The header contains information like sender address, receiver address, type of message, message length and time of sending the message etc. The body of the message can vary in length but should let the total message fit into the capacity of the output and input buffers in the C-Bus interface. The footer contains checksum which aids in the determination of the accuracy of the transmitted message.

At each processor the C-Bus interface supports an interrupt driven message system. An interrupt request is issued when the output buffer is empty or when the input buffer is full. The message control mechanism adopted for the message communication over the C-Bus, at each processor, is as shown in fig.3.2. Further this interrupt driven message system ensures that the output and input buffers, which form the critical resource in the message communication mechanism of CUENET are serviced as quickly as possible so that they are available for passing messages more effectively.
Figure 3.2: Message Control Mechanism in CÜENET
The message communication software, which supports this control mechanism, consists of an interrupt handler, a message transmit module and a message receive module. When a user task requests a message transfer, it invokes the message transmit routine and supplies the address of the receiver (LPN), address of the starting location of the message and the message length. The message transmit module performs the following operations:

1. It determines the physical address of the receiver using the given LPN of the receiver and the access vector. Also it checks whether this processor is allowed access to the receiver processor.

2. If the message to be transmitted is longer than the output buffer size, it divides the message into fragments so that each fragment can fit into the buffer.

3. Then it transfers this message from the memory location indicated by the sender to the transmit message queue. Also it puts the relevant information for forming the header of the message in a header table.

The message receive module is invoked whenever there is a request for receiving a message from another processor by the user's application program. It performs the following operations:

1. It determines if the message is a retransmission request or information for other software process. If it is a retransmission request, it checks the backlog of messages to
see if the required message can be found. If so it places it in the send queue with the destination marked. If the message is not found then it sends an error message to the master computer.

2. If the message is not a retransmission request, then it verifies the check sum. If error is detected it places a request for retransmission in the send queue. If no error is found, it removes the message header/footer and places the message body into the decoded message queue. Then it places an entry into the message log into which user process looks in when it is waiting for a message.

The interrupt handler is activated when either the input buffer becomes full or the output buffer becomes empty. The various operations it performs are:

1. Determines the nature of the interrupt, whether it is due to an empty output buffer or full input buffer.

2. If it is due to an empty output buffer, it moves the next message from the send queue to the output buffer and alters the status register to indicate to the C-Bus controller that a message is waiting for transmission. Also it places a copy of the message in the backlog of messages.

3. If the interrupt is due to a full input buffer, it moves the message to the receive message queue and alters the status register to indicate that the input buffer is empty. This will indicate to the C-Bus controller that it can perform the next message transfer to that receiver.
The role of the C-Bus controller in the message transmission amongst the processors in the CUENET is the most important one. The C-Bus controller has the overall control of the C-Bus and also has the responsibility of safely delivering the message to the receiver from the sender. Besides it should also resolve the contention for the use of the C-Bus if more than one processor requires to send message at the same time. The C-Bus controller employs the daisy chain scheme for resolving the contention for C-Bus. That is, if two processors request the use of the C-Bus for message transmission, then the processor which is electrically nearer to the C-Bus controller gets the use of the C-Bus. Only when this electrically nearer processor does not have any message to transmit, will the processor next to it get the chance to transmit. At each interface there is a mask bit under the control of the C-Bus controller using which the C-Bus controller can alter the priority of that processor for transmitting a message.

The interaction between the C-Bus controller and the various processors starts with the C-Bus controller activating the "Bus grant" line. The processor first in the line in the daisy chain will respond if it has a message to transmit and its mask bit is not set. In that case it prevents further propagation of the Bus grant signal and activates the Bus grant acknowledge signal and also places its address on the interprocessor data bus. The C-Bus
controller then reads the header of the output buffer of the sender and determines the receiver's address. The C-Bus controller determines whether the receiver's input buffer is free. If so, it will initiate the message transfer in bit parallel and byte serial fashion. At the end of the message transmission the C-Bus controller alters the status register of the sender to indicate that its output buffer is empty and the status register of the receiver to indicate that its input buffer is full. This will enable the interrupt to be raised which calls the attention of the particular processor's communication software.

If on other hand, the C-Bus controller determines that the receiver's input buffer is full, it means that the receiver is not ready to receive this message. So the C-Bus controller sets the mask bit in the sender C-Bus interface so that the sender now will propagate the Bus grant signal down the daisy chain. The C-Bus software stores this sender's address in the stack for further processing at a later stage. This masking achieves two things. It enables better utilization of the C-Bus and C-Bus controller by enabling the other processors, whose destination input buffers are free, to send their messages. Also this mask bit ensures that one high priority processor will not "hogg" the C-Bus all the time. The C-Bus controller after initiating the transfer by setting up the various hardware register pointers for the transfer of the message from the
sender's output buffer to the receiver's input buffer, monitors the transmission by checking the parity of the transmitted word. If an error is found, it reinitiates the transfer and after certain number of repeated trials, the C-Bus controller will initiate the master computer for taking corrective action.

The C-Bus controller software, when idle, will empty the stack and unmask the mask bit in the various status registers of the processors whose addresses were in the stack. Then these processors can again ask for the services of the C-Bus controller for transmission of the message in their output buffer.

3.3 Performance measures:

The performance of a DCS is influenced by the way the various processors interact with each other, the characteristics of the interconnection network and the type of workload. The performance measures are the parameters of the DCS with respect to which the performance is assessed.

In CUENET the various processors communicate only by sending/receiving messages over the C-Bus under the control of the C-Bus controller. So a processor in the CUENET can be executing on the application program resident in its memory, be in the process of sending a message over the C-Bus or be in the process of receiving a message. Fig 3.3 gives a state diagram representation of these states. As discussed in section 1.3.1, the performance measures
S1 - Processor Active on the task
S2 - Processor in the process of sending a message
S3 - Processor in the process of receiving a message
S4 - Processor idle after completing the task

Figure 3.3 State diagram of a CUENET Processor

considered are processing power, interconnection network throughput and processor wait time.

The processing power of CUENET is the average number of processors which are busy operating on the application program (state S1 in fig 3.3). Thus, using relation 1.1, the processing power of CUENET can be expressed in the following manner:
Processing power = \( E(N_1) \) .............................. (3.1)
where \( N_1 \) is the number of processors in state S1 (see fig 3.3).

The throughput of C-Bus is another important measure of performance. This will indicate how quickly the C-Bus is transmitting the messages and how quickly they are consumed at the various destinations. The C-Bus throughput, using relation 1.2, can be expressed in the following way:
\[
\text{C-Bus throughput} = \text{Fraction of time C-Bus is busy} \times \text{Rate of message transmission on C-Bus} \] .............................. (3.2)

The amount of time a processor spends in participating in the message communication in CUENET gives an indication how well the system is suitable for running the particular application. This also includes the wait time in the message queue are full or the message has not arrived.

3.4 CUENET Applications:
The various applications that can be run on a distributed processing system depends on the way it can be partitioned so that it can efficiently be run on the distributed system. Further each application may require different interconnection structure for their interaction.

Since the optimal partitioning of a user job for the purpose of concurrent execution is a hard problem, one would want to experiment with different job partitioning and interconnection strategy before arriving at one which best
suites his application. In such cases the flexibility of the interconnection network architecture plays an important role. Thus CUENET is well suited for this purpose.

Some of the applications investigated for CUENET are the quick sort [TAMIR-83] and the numerical computation problems like the Dirichlet problem.

In quick sort the data to be sorted is partitioned equally onto the various processors and the final sort file is obtained either by concatenation or by performing k-way merge sort on these sorted subfiles. Here a star type interconnection architecture is used since the slave processors interact only with the master processor. The quick sort was tried using 2 and 3 processors in the system.

The Dirichlet problem is solved using parallel, synchronous iterative algorithm. Here the entire problem grid is subdivided into n equal sub regions (n is the number of processors) and each sub region is allotted to a processor. Each processor computes the values for the next time step using Gauss-Seidel iterative method. At the end of each iteration the values of bordering points are communicated to the neighbouring processors. This communication technique simulates a grid type (n,n) interconnection, as each processor send and receive data only from its neighbours. The above algorithm has been implemented on 2 processor system and further investigations
are being made to implement this on higher number of processors.

Another feasible application is to implement an office information system [GROSS-82]. Here each processor functions as a workstation of an office and interstation message communication will be carried over the C-Bus.

The parallel processing power of CUENET can be utilized for carrying out linear predictive analysis of speech signals. Since this analysis has to be carried out in real time a pipeline interconnection architecture could be used which can enhance the speed of processing.

These are some of the representative applications of CUENET.

3.5 TOMP—TORino MultiProcessor:

Different types of distributed systems are designed in several educational institutions to mainly study the various aspects of design, implementation and performance of distributed systems. Both loosely coupled and tightly coupled systems have been built. CUENET is a loosely coupled distributed system. TOMP [MARSAN-83] developed at Politecnico of Torino, Torino, Italy is a tightly coupled multiprocessor system. This system is designed around 16 Bit microprocessors and the prototype version comprises of three Zilog Z8001 processors tightly coupled via a global bus and with shared memory. Fig 3.4 shows the two processor
P₁, P₂ - Processors,
PM₁, PM₂ - Private Memories, CM₁, CM₂ - Common Memories,
LB₁, LB₂ - Local Buses, PB₁, PB₂ - Processor Buses.
GB - Global Bus

Figure 3.4 Two Processor Architecture of TOMP

The Global bus called M3BUS is a 30 lines parallel time multiplexed bus in which the address, data and status information are multiplexed (CONTI-81). The exchange of information among various processors is done on a message passing basis. Each processor, requiring to send a message to another, acquires the control of the M3BUS and the
destination processor common memory and deposits the message in that common memory. In TOMP the sending processor is actively involved throughout the message transmission process. In TOMP a typical message transfer to common memory areas consists of three phases: the bus arbitration phase, selection of common memory phase and the final actual data transfer phase. The control mechanism for the bus arbitration is distributed among the processors of TOMP.

CUENET and TOMP illustrate the diverse design techniques that can be exercised in designing a distributed processing system. The two systems differ in many aspects, the fundamental one being the nature of coupling. Further CUENET employs a centrally controlled message communication scheme where as in TOMP the M3BUS global bus arbitration and control is decentralized. The C-Bus is a non-multiplexed parallel bus as compared to M3BUS which is time multiplexed parallel bus. In CUENET one can use a mixture of 8 and 16 bit processors whereas TOMP is predominantly designed for 16 bit processors.

The common feature in both CUENET and TOMP is that both have made use of single board microprocessor system and have added some specialized hardwares to realize functions which are specific to their architectures.

In the performance study of CUENET the graphical modeling techniques SPN and GSPN have been employed. Since
these techniques were also applied in the performance studies of TOMP, we have chosen TOMP for comparing the results of the modeling. From these comparisons we can learn about the versatility of these graphical modeling techniques.
CHAPTER IV

STOCHASTIC PETRI NET (SPN) MODELING OF CUENET.

4.1 Modeling Assumptions:
Here we are concerned with the intercommunication aspects of
CUENET at the message passing level, in order to study how
the input and output buffers of the different computers of
the DCS and the characteristics of the C-Bus controller
affect the overall processing power of the DCS.

The following assumptions are made regarding the
functioning of CUENET:
1) We can clearly identify two periods of activity in each
computer of the DCS, a processing period and a message
transfer period which alternate.
2) The processing period for each processor is assumed to
be an exponentially distributed random variable with a mean
rate $\lambda$. All processors of CUENET are assumed to process at
this mean rate.
3) The message transfer period for the C-Bus controller is
also assumed to be an exponentially distributed random
variable but with mean rate $\mu$.
4) At the end of a processing period each processor in the
DCS may either have a message for transmission or might want
to receive a message. Thus for each processor the message
generation and message reception have equal probability of
occurrence.
5) The processor can send a message to any processor (including itself) with equiprobability i.e., if there are $N$ processors in the system, the probability of sending or receiving a message to or from a processor is $1/N$.

6) For simplifying the modeling, it is further assumed that each processor in the CUENET is allocated one task only.

7) When a processor needs to send a message, if the required buffers, queues and C-Bus are not free, it idles till these resources are available. Similarly when the processor needs to receive a message and its input buffer (queue) does not contain it, then also the processor idles till it receives the required message.

8) The time required for determining the presence of message in the output buffer and checking the availability of the destination input buffer are assumed to be negligibly small when compared to the message transfer time and the processing period. Similarly the release of the output buffer and C-Bus at the end of a message transfer and indication of the destination input buffer being full are assumed to take negligible amount of time.

The assumptions 7 and 8 help in reducing the complexity of the SPN model. The assumptions of exponential distributions for the message generation and message transfer provide Markovian properties to the SPN model and makes the solution of the underlying Markov chain feasible.
4.2 Estimation of system parameters:
The mean rates $\lambda$ and $\mu$ are said to be the system parameters of the SPN model of CUENET. Then the system load factor $\rho$ is given by the ratio $(\lambda/\mu)$. The performance of the system varies with the system's load factor $\rho$. To study this variation in our SPN model we need some representative values for $\lambda$ and $\mu$.

The values of $\mu$ depends on the characteristics of the communication subnet of CUENET. The rate at which the C-Bus controller transfers the message on the C-Bus depends on the speed of the C-bus controller software and the message communication software at each of the processor of CUENET. Let us assume a clock rate of 1 MHz for the C-bus controller and the various processors of the CUENET. The mean length of messages is assumed to be 256 bytes.

The approximate time required for the various stages in the message communication in CUENET are as given below:
(a) The table set up and update time required for the message transmit and interrupt handler part of the message communication software at the sending processor is about 600 $\mu$secs.
(b) Transfer time for the transmit interrupt handler to move the message from the send queue to the output buffer is 60 $\mu$secs/byte.
(c) Overhead time incurred by the C-Bus controller software is 300 $\mu$secs.
(d) Transfer from the sender's output buffer to the receiver's input buffer by the C-Bus controller is 9 μsecs/byte.

(e) The table look up and update overheads for the message receive and interrupt handler part of the message communication software at the receiver requires about 800 μsecs.

(f) The transfer time for the receive interrupt handler to move the message from the input buffer to the receive queue is 15 μsecs/byte.

In order to simplify the SPN modeling of the message communication of CUENET, all these various transfer and processing times are combined together and we get a total message transfer time of 23.6 msecs which gives a rate of 41.66 messages per second.

The values of λs depend on the rate at which messages are generated and this, in reality, depends on the type of application that is run on the system. For the sake of our study we assume different values for the load factor $\rho$ and thus determine the values for λs.

4.3 Two processor SPN model:
The SPN model (SPN2A) in Fig 4.1 models the physical interaction between the two processors PE1 and PE2 and the C-Bus controller during message communication on the C-Bus. In order to simplify the model, in the first attempt,
Figure 4.1 Two Processor SPN model - SPN2A
we have assumed that there are no send and receive queues. The two processors are assumed to have equal priority in using the C-Bus for message transmission. However, since the C-Bus controller employs a daisy chain for bus arbitration, there is a built in priority scheme.

The SPN2A has 15 places and 12 transitions. For modeling purpose, we assume that at the start both the processors are "active", meaning they are working on the task, allotted to them, which are present in their local memory. Also the initial condition is that the C-Bus and output and input buffers of both the processors are free. This initial state is indicated by the token distribution of SPN2A in fig 4.1. Tokens in places $p_1$, $p_4$ and $p_{11}$ indicate that processor PE$_1$ is active and its output and input buffers are free. Similarly tokens in places $p_{10}$, $p_{13}$ and $p_6$ indicate that processor PE$_2$ is active and its output and input buffers are free. The token in place $p_7$ denotes that the C-Bus and C-Bus controller are free.

Let assume that PE$_1$ needs to send a message to PE$_2$. This is modeled by the firing of the enabled transition $t_2$, putting a token in place $p_5$. This enables transition $t_1$, which models both the message transfer to the output buffer by the transmit part of PE$_1$'s interrupt handler and the C-Bus controller operation of checking the availability of the receiver's input buffer and then initiating the message transfer. The firing of the transition $t_1$ puts a token in
places $p_4$ and $p_{14}$ indicating that now the processor $PE_1$ can go on with its work while the message transmission is in progress. The token in place $p_{14}$ enables the transition $t_9$, the firing of which denotes the actual message transfer by the C-Bus controller. The firing of this transition puts tokens in places $p_4$, $p_7$ and $p_8$ indicating that $PE_1$ output buffer and C-Bus are free and message is now present in the input buffer of $PE_2$. The firing of transition $t_6$ models $PE_2$ needing to receive a message, by putting a token in place $p_9$. The tokens in places $p_8$ and $p_9$ enables the transition $t_5$ which models the receive part of the interrupt handler of $PE_2$. The firing of $t_5$ indicates the receipt of message by $PE_2$ sent by processor $PE_1$. This completes one message communication cycle. Similarly the message transmission from processor $PE_2$ to processor $PE_1$ is modeled by transitions $t_7, t_8$ and $t_{10}$ and the reception of this message by $PE_1$ is modeled by $t_3$ and $t_4$.

The transitions $t_{11}$ and $t_{12}$ are included to take care of certain situations which may cause the system to deadlock. In the real system, these situations may be handled by the operating system on each processor. One deadlock situation is when the processors $PE_1$ and $PE_2$ want to receive messages i.e. places $p_2$ and $p_9$ have tokens but neither of the processors have transmitted messages i.e. places $p_3$ and $p_8$ do not have any token. Transition $t_{11}$ resolves this situations in the SPN model, by putting both processors into
active state.

Another deadlock situation occurs when both processors want to transmit a message while their destination input buffers are not free. Transition \( t_{12} \) detects this situation and resolves it by putting the two processors into active state at the same time the tokens in places \( p_3 \) and \( p_8 \) are retained.

The modeling assumptions discussed in section 4.1 determine the rates at which the various transitions fire in a given SPN model. The transitions \( t_2, t_3, t_6 \) and \( t_7 \), which model the processors generation reception of messages fire at the rate \( \lambda/2 \). The transitions \( t_9 \) and \( t_{10} \), which model the transmission of message by the C-Bus controller, fire at the rate \( \mu \). The transitions \( t_1, t_8, t_4, t_5, t_{11} \) and \( t_{12} \), which model the activities like determination of the availability of the C-Bus, receiver's input buffer, receipt of the message by the destination processor etc, fire at a very high rate, so as to minimize their effect on the system parameters and performance measures.

SPN2A will give a pessimistic estimate of the processing power. This is because the transition \( t_1 \) (\( t_8 \)) model the C-Bus controller selection function and the message transfer to the output buffer by the sender processor's interrupt handler. Consider the enabling conditions for transition \( t_1 \). It cannot fire till all its input places \( p_4, p_5, p_6 \) and
P7 contain a token. Thus in a situation, where P4, P5 and P7 have tokens and P6 does not have a token which indicates that the input buffer of PE2 is not free, the processor PE1 is held in wait state, till the time PE2 receives the message in the input buffer and frees it. But in actual case this does not happen, since the processor PE1 just checks if its output buffer is free and if so puts the message in it and returns to its active state.

In order to overcome this drawback we modified this SPN to model these actions separately.

4.4 Modified two processor SPN model:
The SPN model (SPN2B) shown in fig 4.2, models the C-Bus controller selection function in more detail, by using an additional place and an additional transition for each processor. Thus it has 17 places and 14 transitions. The transitions t1 and t2 model the selection function that was modeled by t1 in SPN2A. Here transition t2 models the message movement to the output buffer by processor PE1's interrupt handler and transition t1 models the C-Bus controller selection and message transfer initiation. Similarly the transitions t10 and t11 model the functions modeled by the transition t8. Transition t2 gets enabled when the places P3 and P4 have tokens i.e. PE1 has generated a message and its output buffer is free. The transition t2 fires by putting a token in places P1 and P5 i.e. the message is in its output buffer and PE1 returns to active
Figure 4.2 Modified Two Processor SPN model - SPN2B
state. The transition \( t_1 \) gets enabled only when there are tokens in places \( p_1, p_2 \) and \( p_9 \) i.e. a message is in PE output buffer, PE2 input buffer and C-Bus are free. The other part of message communication is modeled in the same way as done in SPN2A. This improvement models the concurrency in the system more closely.

The transitions \( t_3, t_2, t_1 \) and \( t_6 \) model the message transmission from PE1 to PE2 while the message transmission from PE2 to PE1 is modeled by transitions \( t_9, t_{10}, t_{11} \) and \( t_{12} \).

The message reception by PE1 is modeled by transitions \( t_4 \) and \( t_5 \), while transitions \( t_8 \) and \( t_7 \) model the message reception by PE2.

Transitions \( t_{13} \) and \( t_{14} \) are similar to transitions \( t_{11} \) and \( t_{12} \) of SPN2A, for resolving the two deadlock situations.

In SPN2B, similar to SPN2A, the transitions \( t_3, t_4, t_8 \) and \( t_9 \) fire at the rate \( \lambda/2 \). The transitions \( t_6, t_{12} \) modeling message transmission fire at the rate \( \mu \). The rest of the transitions modeling availability of the output buffer, receiver input buffer etc. fire at a very high rate.

4.5 Three Processor SPN Model:
Fig4.3 shows the three processor SPN model of CUENET(SPN3A). SPN3A is on the same lines as SPN2A shown in fig4.1.

In this SPN model also, for the sake of simplifying the model, we assume that there are no transmit and receive
Figure 4.3 Three Processor SPN model - SPN3A
message queues at each processor. Each processor has equal priority in using the C-Bus and C-Bus controller for sending a message. Also each processor sends messages to the other with equal probability.

SPN3A comprises of 25 places and 23 transitions. We assume, initially all the processors are "active" working on the program segment allotted to them in their memory. The C-Bus and the various output and input buffers are free. This initial condition is represented by the token distribution shown in fig 4.3.

Let us assume that PE₁ needs to send a message to either PE₂ or PE₃. This is indicated by the firing of the enabled transition t₁₂, which moves the token in place p₁₇ to the place p₁₃. Now the transitions t₈ and t₁₃ are both simultaneously enabled, provided that the places p₈,p₁₂,p₁₄ and p₁₅ have tokens in them. Here transition t₈ models the message transfer to the output buffer by PE₁'s interrupt handler and the C-Bus selection function for message transfer between PE₁ and PE₂. Similarly t₁₃ models these functions for message transfer from PE₁ to PE₃. Since we have equal probability of sending a message to either processor PE₂ or PE₃, let us assume that the transition t₈ fires, thus routing the message to processor PE₂. The firing of transition t₈ puts a token in places p₁₇(PE₁ active) and place p₅, removes one token each from places p₈,p₁₂,p₁₃ and p₁₄ respectively. Transition t₅ is enabled
and firing of which (indicates the completion of message transfer by the C-Bus controller to the processor PE₂. The firing of transition t₅ removes a token from place p₅ and puts one token each at places p₆, p₁₂ and p₁₄ indicating that PE₂ input buffer contains the message and the output buffer of PE₁ and C-Bus are free. The firing of enabled transition t₆, removes the token from place p₄ and puts a token in place p₇ indicating that PE₂ wants to receive a message. Now places p₆ and p₇ contain tokens thus enabling transition t₇ which models the function of removing the message from the input buffer by the interrupt handler of PE₂. The firing of transition t₇ removes tokens from places p₆, p₇ and puts tokens in places p₄, p₈. This indicates the reception of message by processor PE₂ and completes one message communication cycle.

The following is the relationship between the group of transitions and the pairs of processors from the message transmission point of view.

<table>
<thead>
<tr>
<th>Transitions</th>
<th>Processor (from, to)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(t₁₂, t₁₃, t₁₄)</td>
<td>(PE₁, PE₃)</td>
</tr>
<tr>
<td>(t₃, t₁, t₂)</td>
<td>(PE₂, PE₁)</td>
</tr>
<tr>
<td>(t₃, t₄, t₉)</td>
<td>(PE₂, PE₃)</td>
</tr>
<tr>
<td>(t₁₅, t₁₆, t₂₀)</td>
<td>(PE₃, PE₁)</td>
</tr>
<tr>
<td>(t₁₅, t₁₁, t₁₀)</td>
<td>(PE₃, PE₂)</td>
</tr>
</tbody>
</table>

The message reception by PE₃ and PE₁ are modeled by
transitions $t_{18}, t_{21}$ and $t_{17}, t_{19}$ respectively. The transitions $t_{22}$ and $t_{23}$ are provided to do the same operations as done by the transitions $t_{11}$ and $t_{12}$ respectively in the model SPN2A.

The firing rates of the various transitions are similar to the firing rates of transitions of SPN2A. The transitions $t_6, t_7, t_{12}, t_{15}, t_{17}$ and $t_{18}$ fire at the rate $\lambda/2$ and transitions $t_2, t_5, t_9, t_{10}, t_{14}$ and $t_{20}$ fire at the rate $\mu$. The rest of the transitions which model the operations like determination of the availability of the receiver, input buffer by the C-Bus controller etc. fire at a very high rate.

4.6 Modified Three processor SPN model:

The SPN model shown in fig 4.4 gives the three processor SPN model (SPN3B) similar to the model SPN2B. SPN3B has 28 places and 26 transitions. The following is the correspondence between the group of transitions and the pairs of processors, from the message transmission point of view:

<table>
<thead>
<tr>
<th>Transitions</th>
<th>Processors (from,to)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(t_{14}, t_{12}, t_8, t_3)$</td>
<td>$(PE_1, PE_2)$</td>
</tr>
<tr>
<td>$(t_{14}, t_{12}, t_{15}, t_{16})$</td>
<td>$(PE_1, PE_3)$</td>
</tr>
<tr>
<td>$(t_6, t_7, t_2, t_1)$</td>
<td>$(PE_2, PE_1)$</td>
</tr>
<tr>
<td>$(t_6, t_7, t_10, t_{17})$</td>
<td>$(PE_2, PE_3)$</td>
</tr>
<tr>
<td>$(t_{18}, t_{13}, t_{11}, t_{23})$</td>
<td>$(PE_3, PE_1)$</td>
</tr>
<tr>
<td>$(t_{18}, t_{13}, t_{24}, t_9)$</td>
<td>$(PE_3, PE_2)$</td>
</tr>
</tbody>
</table>
From the message reception modeling aspect, the following is the relationship between the transitions and the corresponding processors:

Transition | Processor
---|---
\(t_{19}, t_{21}\) | PE_1
\(t_5, t_4\) | PE_2
\(t_{20}, t_{22}\) | PE_3

The transitions \(t_{25}\) and \(t_{26}\) function similar to transitions \(t_{13}\) and \(t_{14}\) in SPN2B, resolving the two deadlock situations.

In this SPN model the transitions \(t_5, t_6, t_{14}, t_{18}, t_{19}\) and \(t_{20}\) fire at the rate \(\lambda/2\) and transitions \(t_1, t_3, t_9, t_{16}, t_{17}\) and \(t_{23}\) fire at the rate \(\mu\). The rest of the transitions modeling the operations like checking the availability of the output buffer, availability of receiver's input buffer, etc. fire at a very high rate.

4.7 Analysis of SPN models:

Determination of Reachability set:

In chapter 2, we have discussed the two methods, for the determination of reachability set— one using prime numbers and the other using binary and hexadecimal numbers.

The prime number method is employed in determining the reachability set for the two SPN models SPN2A and SPN2B. Since the prime number method has a limitation on the number of places due to the reasons stated in section 2.2.2, the
reachability sets for the three processor models SPN3A and SPN3B are determined using the binary number method.

As discussed in section 2.2.2, the presence of a "terminal" marking in the reachability set indicates the presence of a deadlock situation within the system being modeled. Thus during the process of determination of the reachability set, these methods also check for the presence of a terminal marking in the reachability set generated.

Table 4.1 gives the number of markings in each reachability set of these SPN models discussed. A representative listing of the various markings in a particular reachability set is given in Appendix D. This is the reachability set for the two processor model SPN2A.

Determination of the steady state probabilities of the states of the underlying Markov chain:

To determine the steady state probabilities of the states of an ergodic Markov chain, the transition rate matrix Q must be formed first. The order of the transition matrix Q is the number of states present in the underlying Markov chain. A nonzero entry q_{ij} in row i and column j in the transition matrix Q indicates that the system being in state i will move to state j at a rate of q_{ij}. This movement from state i to state j is represented in a SPN model in the following way. The state i represents a distribution of tokens in the SPN such that a transition is enabled firing of which
## TABLE 4.1

DETAILS OF MODEL AND REACHABILITY SET SIZE FOR SPN MODELS OF CUNET.

<table>
<thead>
<tr>
<th>SPN MODEL</th>
<th>NO. OF PLACES</th>
<th>NO. OF TRS</th>
<th>SIZE OF R.S.</th>
<th>SPARSITY OF TR. MATX</th>
<th>NZ. ELES</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SPN2A</td>
<td>15</td>
<td>12</td>
<td>72</td>
<td>311</td>
<td>6.0</td>
<td></td>
</tr>
<tr>
<td>SPN2B</td>
<td>17</td>
<td>14</td>
<td>216</td>
<td>991</td>
<td>2.12</td>
<td></td>
</tr>
<tr>
<td>SPN3A</td>
<td>25</td>
<td>23</td>
<td>864</td>
<td>4578</td>
<td>0.61</td>
<td></td>
</tr>
<tr>
<td>SPN3B</td>
<td>28</td>
<td>26</td>
<td>4320</td>
<td>27995</td>
<td>0.15</td>
<td></td>
</tr>
</tbody>
</table>

**NOTE:**
- TRS.: Transitions.
- TR. MATX: Transition Matrix.
- NZ. ELES: Non-zero Elements.
- R S.: Reachability Set.
produces a new marking represented by state \( j \). The value of the nonzero entry \( q_{ij} \) in the transition matrix \( Q \) corresponds to the rate at which this enabled transition \( t \) fires. In a particular marking (state) \( i \) more than one transition may be enabled. This is represented by the number of nonzero elements in the \( i \) th row of \( Q \). The diagonal element \( q_{ii} \) is obtained by the relation,

\[
\sum_{j \neq i} q_{ij} = -q_{ii} \tag{4.1}
\]

This relation makes the sum of \( q_{ij} \)s along a row equals to zero and also makes the transition matrix \( Q \) diagonally dominant which gives it good numerical stability [KLEIN-75, MEDHI-82]. The transition matrix \( Q \) has a large number of zero entries thus making it highly sparse. This has a bearing on the way it can be solved.

After the formation of the transition matrix \( Q \), the steady state probabilities of the various states of the Markov chain are determined by solving the equation,

\[
Q^{T}y^{T} = 0 \tag{4.2}
\]

The vector \( y^{T} \) is a column vector which gives the steady state probability of each state in the ergodic Markov chain.

As discussed in section 2.3.3, the equation 4.2 is converted to \( Ax=b \) form and then solved. The solution can be obtained using any of the methods of solving a linear system of simultaneous equations. In the initial investigation of this thesis the Gauss-Jordan elimination method with optimum pivoting was used.
Determination of the values of the Performance measures:

From the steady state probabilities the token densities of the places of the SPN model are determined. As discussed in section 2.3.2, the token density of a place $p_i$ is given by the relation,

$$d_i = \sum_{j \in R(PN)} y_j \cdot m_{ij} \quad (4.3)$$

where $y_j$ is the steady state probability of state $j$ and $m_{ij}$ is the number of tokens in place $p_i$ in state $j$.

Once the token densities of the various places of the SPN model are determined the various performance measures are obtained as shown below.

Processing Power $P = \text{Sum of the token densities of the places which correspond to the active processors in the DCS}$ (see relation 3.1).

Example, for SPN2A, Processing power = $d_1 + d_{10}$.

C-Bus Throughput = $(1\text{-tokendensity of the place which models the free condition of C-Bus}) \times \mu$ (see relation 3.2).

Example, In SPN2A, C-Bus Throughput = $(1-d_7) \times \mu$.

Processor wait time = sum of token densities of places representing message for transmission and message receive request by that processor.

Example, in SPN2A, $PE_1$ wait time = $d_2 + d_5$.

- **4.8 Discussion of SPN models of CUENET:**

The table 4.1 shows how the reachability set increases with the number of places and number of transitions. In this table we observe that in case of two processor SPN models,
SPN2B differs from SPN2A in having 2 extra places and 2 extra transitions. But the number of markings of SPN2B is 300% of that of the markings of SPN2A. This indicates an enormous growth in the number of markings in the SPN with a small increase in the number of places and transitions. The direct consequence of this is the increased time complexity in obtaining the steady state probabilities. The sparsity of the rate matrix $Q$ is also shown in Table 4.1. The non-zero elements of $Q$ do not exhibit any structure that are well known to sparse matrix techniques such as band matrix, tridiagonal matrix etc. Hence in the initial stages of this thesis, Gauss-Jordan elimination method was adopted for obtaining the solution to equation 4.2.

Table 4.2 indicates the time requirements for obtaining the solution by this method for the SPN models considered. The estimated time requirements for SPN3A is done with maintaining the data file externally, since the complete matrix cannot be resident in the central memory due to the restriction in its availability. Hence the difficulty in obtaining the solution to SPN3A and SPN3B.

The reachability set of a loosely coupled system is relatively large. This becomes evident from the following comparison. Marsan et al [MARSAN-83] have employed SPNs in their performance analysis of the TOMP multiprocessor (see section 3.5). Table 4.3 gives a comparison of the two SPN models (i.e. CUENET and TOMP) in terms of the number of
### TABLE 4.2

**COMPUTATION TIMES FOR SPN ANALYSIS:**

<table>
<thead>
<tr>
<th>SPN MODEL</th>
<th>SIZE OF</th>
<th>R S</th>
<th>T1</th>
<th>T2</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPN2A</td>
<td>72</td>
<td>0.233</td>
<td>14.1</td>
<td></td>
</tr>
<tr>
<td>SPN2B</td>
<td>216</td>
<td>1.340</td>
<td>367.0</td>
<td></td>
</tr>
<tr>
<td>SPN3A</td>
<td>864</td>
<td>29.360</td>
<td>6000.0 *</td>
<td></td>
</tr>
</tbody>
</table>

**Note:**
- **T1**: Time in secs for determining the Reachability set and form Transition rate Matrix.
- **T2**: time in secs to obtain the solution from the Transition rate Matrix.
- **R S**: Reachability Set.
- *****: Estimated Value

### TABLE 4.3

**COMPARISION OF CUENET AND TOMP SPN MODELS:**

<table>
<thead>
<tr>
<th>CUENET</th>
<th>TOMP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>SPN MODEL</td>
<td>NO. OF PLACES</td>
</tr>
<tr>
<td>SPN2A</td>
<td>15</td>
</tr>
<tr>
<td>SPN3A</td>
<td>25</td>
</tr>
</tbody>
</table>

**Note:**
- **TRS.**: Transitions.
- **R S**: Reachability Set.
SPN2B differs from SPN2A in having 2 extra places and 2 extra transitions. But the number of markings of SPN2B is 300% of that of the markings of SPN2A. This indicates an enormous growth in the number of markings in the SPN with a small increase in the number of places and transitions. The direct consequence of this is the increased time complexity in obtaining the steady-state probabilities. The sparsity of the rate matrix \( Q \) is also shown in Table 4.1. The non-zero elements of \( Q \) do not exhibit any structure that are well known to sparse matrix techniques such as band matrix, tridiagonal matrix etc. Hence in the initial stages of this thesis, Gauss-Jordan elimination method was adopted for obtaining the solution to equation 4.2.

Table 4.2 indicates the time requirements for obtaining the solution by this method for the SPN models considered. The estimated time requirements for SPN3A is done with maintaining the data file externally, since the complete matrix cannot be resident in the central memory due to the restriction in its availability. Hence the difficulty in obtaining the solution to SPN3A and SPN3B.

The reachability set of a loosely coupled system is relatively large. This becomes evident from the following comparison. Marsan et al [MARSAN-83] have employed SPNs in their performance analysis of the TOMP multiprocessor (see section 3.5). Table 4.3 gives a comparison of the two SPN models (i.e. CUENET and TOMP) in terms of the number of
TABLE 4.2

COMPUTATION TIMES FOR SPN ANALYSIS:

<table>
<thead>
<tr>
<th>SPN MODEL</th>
<th>SIZE OF R S</th>
<th>T1</th>
<th>T2</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPN2A</td>
<td>72</td>
<td>0.233</td>
<td>14.1</td>
</tr>
<tr>
<td>SPN2B</td>
<td>216</td>
<td>1.340</td>
<td>367.0</td>
</tr>
<tr>
<td>SPN3A</td>
<td>864</td>
<td>29.360</td>
<td>6000.0</td>
</tr>
</tbody>
</table>

Note:
- T1: Time in secs for determining the Reachability set and form Transition rate Matrix.
- T2: Time in secs to obtain the solution from the Transition rate Matrix.
- R S: Reachability Set.
- *: Estimated Value

TABLE 4.3

COMPARISON OF CUENET AND TOMP SPN MODELS:

<table>
<thead>
<tr>
<th>CUENET</th>
<th>TOMP</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPN</td>
<td>NO.OF MODEL</td>
</tr>
<tr>
<td>SPN2A</td>
<td>15</td>
</tr>
<tr>
<td>SPN3A</td>
<td>25</td>
</tr>
</tbody>
</table>

Note:
- TRS.: Transitions.
- R S: Reachability Set.
places and transitions and size of the reachability sets for the case of two and three processors. The two processor SPN model of TOMP has more number of transitions as compared to the two processor CUENET model (SPN2A) while both have the same number of places. We observe that the two processor TOMP SPN has only 26 markings in its reachability set whereas the SPN2A has 76 states. This large difference is seen in the case of three processor models also. It is interesting to note that SPN3A has lesser number of places and transitions as compared to the TOMP model.

This increase in the number of states is attributable to the loose coupling nature of CUENET. A loosely coupled system exhibits more asynchronism. This enables the system components to function more independently. Thus an SPN which models these actions will have more number of states in its reachability set.

The results of the SPN analysis of the two processor models SPN2A and SPN2B are presented in Tables 4.4 and 4.5 respectively. These tables list the variation of the processing power, C-Bus throughput and the processor wait time with the system load factor \( \rho \). From these results it can be seen that SPN2B gives a more optimistic results as compared to SPN2A, as discussed earlier in this chapter.

The large growth of the number of states in the reachability set and the consequent time complexity in the
TABLE 4.4

SPN ANALYSIS RESULTS OF SPN2A.

<table>
<thead>
<tr>
<th>( \rho )</th>
<th>( \lambda )</th>
<th>( \mu )</th>
<th>( PP )</th>
<th>( CT )</th>
<th>( P1W )</th>
<th>( P2W )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>0.417</td>
<td>41.66</td>
<td>1.4626</td>
<td>0.2249</td>
<td>0.2687</td>
<td>0.2687</td>
</tr>
<tr>
<td>0.05</td>
<td>2.803</td>
<td>41.66</td>
<td>1.4588</td>
<td>1.1082</td>
<td>0.2706</td>
<td>0.2706</td>
</tr>
<tr>
<td>0.1</td>
<td>4.166</td>
<td>41.66</td>
<td>1.4530</td>
<td>2.2080</td>
<td>0.2736</td>
<td>0.2736</td>
</tr>
<tr>
<td>0.2</td>
<td>8.333</td>
<td>41.66</td>
<td>1.4376</td>
<td>4.3409</td>
<td>0.2812</td>
<td>0.2812</td>
</tr>
<tr>
<td>0.4</td>
<td>16.666</td>
<td>41.66</td>
<td>1.3942</td>
<td>8.3195</td>
<td>0.3030</td>
<td>0.3030</td>
</tr>
<tr>
<td>0.6</td>
<td>24.996</td>
<td>41.66</td>
<td>1.3384</td>
<td>11.8648</td>
<td>0.3307</td>
<td>0.3307</td>
</tr>
<tr>
<td>0.8</td>
<td>33.328</td>
<td>41.66</td>
<td>1.2766</td>
<td>14.9601</td>
<td>0.3617</td>
<td>0.3617</td>
</tr>
<tr>
<td>1.0</td>
<td>41.66</td>
<td>41.66</td>
<td>1.2130</td>
<td>17.6347</td>
<td>0.3965</td>
<td>0.3965</td>
</tr>
</tbody>
</table>

**NOTE:**

- **PP**: Processing Power.
- **CT**: C-Bus Throughput in messages/sec.
- **P1W**: Wait time for PE1.
- **P2W**: Wait time for PE2.
### Table 4.5

SPN Analysis Results of SPN2B.

<table>
<thead>
<tr>
<th>$\rho$</th>
<th>$\lambda$</th>
<th>$\mu$</th>
<th>PP</th>
<th>CT</th>
<th>PLW</th>
<th>P2W</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>0.417</td>
<td>41.66</td>
<td>1.5406</td>
<td>0.2625</td>
<td>0.2297</td>
<td>0.2297</td>
</tr>
<tr>
<td>0.05</td>
<td>2.083</td>
<td>41.66</td>
<td>1.5342</td>
<td>1.3040</td>
<td>0.2329</td>
<td>0.2329</td>
</tr>
<tr>
<td>0.1</td>
<td>4.166</td>
<td>41.66</td>
<td>1.5246</td>
<td>2.5871</td>
<td>0.2377</td>
<td>0.2377</td>
</tr>
<tr>
<td>0.2</td>
<td>8.333</td>
<td>41.66</td>
<td>1.5014</td>
<td>5.0825</td>
<td>0.2492</td>
<td>0.2492</td>
</tr>
<tr>
<td>0.4</td>
<td>16.666</td>
<td>41.66</td>
<td>1.4408</td>
<td>9.6860</td>
<td>0.2796</td>
<td>0.2796</td>
</tr>
<tr>
<td>0.6</td>
<td>24.996</td>
<td>41.66</td>
<td>1.3674</td>
<td>13.7061</td>
<td>0.3163</td>
<td>0.3163</td>
</tr>
<tr>
<td>0.8</td>
<td>33.328</td>
<td>41.66</td>
<td>1.2890</td>
<td>17.1306</td>
<td>0.2890</td>
<td>0.2890</td>
</tr>
<tr>
<td>1.0</td>
<td>41.66</td>
<td>41.66</td>
<td>1.2104</td>
<td>20.0051</td>
<td>0.3949</td>
<td>0.3949</td>
</tr>
</tbody>
</table>

**NOTE:**

- PP: Processing Power.
- CT: C-Bus Throughput in Messages/sec.
- PLW: Wait time for PE1.
- P2W: Wait time for PE2.
mathematical solution limits the use of SPN in modeling CUENET. This prompted us to consider the Generalized Stochastic Petri nets (GSPN), which are derivatives of the SPNs for the modeling of CUENET.
CHAPTER V

GENERALIZED STOCHASTIC PETRINET (GSPN) MODELING OF CUENET.

5.1 Preliminaries:
In chapter IV the SPN modeling of CUENET has brought home the fact that the increase in the solution complexity, when we progressed from two to three processors, is because of the enormous increase in the number of states in the reachability set. In an effort to limit the size of the reachability set and at the same time maintain reasonable level of details in the model, we have adopted the GSPN (proposed by Marsan et al) [MARSAN-84] for the modeling of message communication in CUENET.

In the GSPN modeling of CUENET, the assumptions made for the SPN models still hold. Also we are employing the same values of $\lambda$s and $\mu$s as used in the SPN models.

Since GSPN has two kinds of transitions, namely timed and immediate, in our models we have adopted the following assignment rules for the various transitions. Those transitions which model message generation, transmission and reception are treated as transitions. Each of them fire at a mean rate following an exponential distribution. The system parameters $\lambda$ and $\mu$ determine these mean rates. The other transitions which model the various other operations of CUENET like checking the availability of the different
resources required, resolving deadlock situations are represented as immediate transitions which fire in zero time.

5.2 Two processor GSPN models:

Fig 5.1 shows the two processor GSPN model (GSPN2A) of CUENET which is similar to the SPN2A. The assumptions made for simplifying the SPN2A also hold good for GSPN2A (See page No. 87). The GSPN2A has 15 places, 6 timed transitions and 6 immediate transitions. The transitions $t_2, t_3, t_6, t_7, t_9$ and $t_{10}$ are timed transitions while the transitions $t_1, t_4, t_5, t_8, t_{11}, t_{12}$ are immediate transitions.

The interpretations for these transitions are exactly similar to that of SPN2A. The main difference of GSPN is the way in which the various immediate transitions are assigned probabilities of firing when more than one immediate transitions get enabled in a given marking. The firing probability for each immediate transition depends on the number of immediate transitions that get enabled in a particular marking and on certain characteristics of the system being modeled. These various probabilities are given in the form of a probability distribution called "switching distribution".

In this model, the immediate transitions $t_1, t_4$ cannot get enabled at the same time, due to the conflicting timed transitions $t_2$ and $t_3$. For the same reason, the immediate
Figure 5.1 Two processor GSPN model - GSPN2A
transitions $t_5$ and $t_8$ cannot get enabled simultaneously.

The immediate transitions $t_1$ and $t_5$ cannot get enabled at the same instant, due to the reason places $p_6$ and $p_8$ cannot have tokens at the same time. This reason also holds for transitions $t_4$ and $t_8$.

The immediate transitions $t_{11}$ and $t_{12}$, being transitions for resolving deadlock situations, get enabled independently.

Due to the above reasons, only one of these six immediate transitions get enabled in any given vanishing state. Hence their switching distribution is that each of them will fire with a probability 1.0, when they are enabled.

Fig5.2 shows the modified two processor GSPN model (GSPN2B). similar to the SPN2B. The GSPN2B has 17 places, 6 timed and 8 immediate transitions. Once again the interpretations for these timed and immediate transitions are similar to that of the SPN2B.

The switching distributions for the various immediate transitions are as described below:

In this model, like in GSPN2A, immediate transitions $t_2$ and $t_5$ cannot get enabled at the same time due to the two conflicting timed transitions $t_3$ and $t_4$. For the same reason, $t_7$ and $t_{10}$ cannot get enabled at the same time.
Figure 5.2 Modified Two Processor GSPN model - GSPN2B
The immediate transitions \( t_1 \) and \( t_7 \) cannot get enabled at the same time since the places \( p_2 \) and \( p_8 \) cannot contain a token each at the same time (token \( p_2 \) indicates that the particular input buffer free while a token in \( p_8 \) indicates the input buffer contains a message). The immediate transitions \( t_5 \) and \( t_{11} \) cannot get enabled at the same time, for a similar reason.

Thus it is observed, from the various vanishing states, that the transitions \( t_1, t_5 \) and \( t_{10} \) get enabled individually or in pairs. Also the transitions \( t_2, t_7 \) and \( t_{11} \) get enabled individually or in pairs. Transitions \( t_1 \) and \( t_{11} \) get enabled together in certain vanishing states of the GSPN. The switching probability for each of these immediate transitions will include all these possibilities.

The following are the switching probabilities for the immediate transitions:

\( t_1 \): when \( t_5 \) or \( t_{10} \) or \( t_{11} \) is enabled: \( \text{prob} = 1/2 \);
Otherwise, \( \text{Prob} = 1 \).

\( t_2 \): when \( t_7 \) or \( t_{11} \) is enabled: \( \text{Prob} = 1/2 \);
Otherwise: \( \text{Prob} = 1 \).

\( t_5 \): when \( t_1 \) or \( t_{10} \) is enabled: \( \text{Prob} = 1/2 \);
Otherwise: \( \text{Prob} = 1 \).

\( t_7 \): when \( t_2 \) or \( t_{11} \) is enabled: \( \text{Prob} = 1/2 \);
Otherwise: \( \text{Prob} = 1 \).

\( t_{10} \): when \( t_1 \) or \( t_5 \) is enabled: \( \text{Prob} = 1/2 \);
Otherwise: \( \text{Prob} = 1 \).
\[ t_{11} \text{ : when } t_1 \text{ or } t_2 \text{ or } t_7 \text{ is enabled: } \text{Prob } = 1/2; \]
Otherwise : \( \text{Prob} = 1. \)

As we have found the size of the tangible states in the reachability sets were not too large, a more detailed GSPN was evolved. In the case of GSPN2A and GSPN2B models there are no send and receive message queues. In the new model we have included the send and receive queues at each processor and the resultant GSPN model (GSPN2C) is shown in fig. 5.3. The GSPN2C has 23 places, 6 timed and 10 immediate transitions.

The initial state of the model is assumed to be that both the processors \( PE_1 \) and \( PE_2 \) are "active", the various send and receive message queues are empty and the respective input and output buffers and the C-Bus are free. This initial state is indicated by the token distribution shown in fig 5.3. The token \( p_7 \) indicates that \( PE_1 \) is "active" and tokens in \( p_2 \) and \( p_{20} \) represent the output and input buffers of \( PE_1 \) are free. The tokens in \( p_5 \) and \( p_{10} \) indicate the size of the send and receive queues. The places \( p_{17}, p_{22}, p_{14}, p_{19} \) and \( p_{14} \) correspond to the processor \( PE_2 \) and have the same interpretation. The token in place \( p_{12} \) represents that the C-Bus and C-Bus controller are free.

In this model, except for the transitions which model the send and receive queues at each processor, the operation of the other transitions are exactly similar to the ones in GSPN2A.
Figure 5.3 Two processor (with message queues)

GSPN model - GSPN2C.
The immediate transitions $t_2$ and $t_6$ represent the message queueing at the send and receive points in processor PE$_1$. Similarly, $t_{13}$ and $t_9$ model the message queueing in PE$_2$.

The message generation by PE$_1$ and the transfer of the message to PE$_2$ by the C-Bus controller are modeled by the sequential firings of timed transition $t_3$, immediate transitions $t_2,t_1$, timed transition $t_8$ and the immediate transition $t_9$. The message reception by PE$_2$ is modeled by the timed transition $t_{11}$ and immediate transition $t_{10}$.

Similarly the message communication from PE$_2$ to PE$_1$ is modeled by timed transitions $t_{12},t_7$ and immediate transitions $t_{13},t_{14}$ and $t_6$. The reception of the message by PE$_1$ is modeled by timed transition $t_4$ and immediate transition $t_5$.

The immediate transitions $t_{15}$ and $t_{16}$ resolve the two deadlock situations similar to the ones in SPN2A and SPN2B. Since in GSPN2C we are employing multiple tokens, the use of inhibitor arcs simplify the modeling of these deadlock situations.

The switching distributions for each of the immediate transitions are as given below. Here, like in GSPN2A, certain immediate transitions do not get enabled at the same time. They are $(t_2,t_5),(t_{10},t_{13}),(t_1,t_9)$, and $(t_6,t_{14})$. Also the transitions $t_{15}$ and $t_{16}$ get enabled separately.
With these exceptions, the immediate transitions in GSPN2B get enabled singly, in pairs or in triples.

For example, \( t_1 \) get enabled together with \( t_2 \) or \( t_5 \) or \( t_6 \) or \( t_{13} \) or \( t_{14} \) or \( t_{10} \) and in combinations of \( (t_{14}, t_{10}) \) and \( (t_5, t_{14}) \). We will describe the switching distributions of transitions \( t_1, t_2, t_5 \) and \( t_6 \) associated with \( PE_1 \). In a similar manner, from the homogeneous nature of these processors, the switching distributions for transitions \( t_{14}, t_{13}, t_{10} \) and \( t_9 \) can be defined.

Switching distributions for:

\( t_1 \): when \( t_2 \) is enabled: \( \text{Prob} = \frac{m_3}{(m_3 + m_5)} \);
when \( t_5 \) or \( t_6 \) or \( t_{10} \) or \( t_{13} \) or \( t_{14} \) is enabled: \( \text{prob} = \frac{1}{2} \);
when both \( t_{14} \) and \( t_{10} \) or \( t_5 \) and \( t_{14} \) are enabled: \( \text{Prob} = \frac{1}{3} \);
otherwise: \( \text{prob} = 1 \).

\( t_2 \): when \( t_1 \) is enabled: \( \text{Prob} = \frac{m_5}{(m_3 + m_5)} \);
when \( t_6 \) or \( t_{10} \) or \( t_{14} \) is enabled: \( \text{prob} = \frac{1}{2} \);
otherwise \( \text{prob} = 1 \).

\( t_5 \): when \( t_6 \) is enabled: \( \text{prob} = \frac{m_9}{(m_9 + m_{10})} \);
when \( t_{13} \) or \( t_{14} \) is enabled: \( \text{prob} = \frac{1}{2} \);
when both \( t_1 \) and \( t_{14} \) are enabled: \( \text{prob} = \frac{1}{3} \);
otherwise: \( \text{prob} = 1 \).

\( t_6 \): when \( t_1 \) or \( t_2 \) is enabled: \( \text{prob} = \frac{1}{2} \);
when \( t_5 \) is enabled: \( \text{prob} = \frac{m_{10}}{(m_9 + m_{10})} \);
otherwise: \( \text{prob} = 1 \).

In the GSPN2C we model the effect of different message capacity for the message queues by simply altering the
number of tokens in places which model them. For example, places $P_5, P_{10}, P_{14}$ and $P_{19}$ in Fig 5.3 contain 2 tokens each indicating that this GSPN models a two processor CUENET with size of the message queue 2. To study the effects with message queue size 3 messages we just alter the number of tokens in these places to 3.

5.3 Three processor GSPN model:
Fig 5.4 shows the three processor GSPN model of CUENET(GSPN3) which is similar to the three processor model SPN3A. GSPN3 has 25 places, 12 timed transitions and 11 immediate transitions.

The timed transitions $t_{12}, t_3$ and $t_{15}$ model the message generation by processors $PE_1, PE_2$ and $PE_3$ respectively. The message transmission from $PE_1$ to $(PE_2, PE_3)$, from $PE_2$ to $(PE_1, PE_3)$ and from $PE_3$ to $(PE_1, PE_2)$ are modeled by the pairs of timed transitions $(t_5, t_{14}), (t_2, t_9)$ and $(t_{20}, t_{10})$ respectively. The message reception by $PE_1$ is modeled by the timed transition $t_{17}$ and the immediate transition $t_{19}$. In the similar manner, the timed and immediate transition pairs $(t_6, t_7)$ and $(t_{18}, t_{21})$ model the message reception by $PE_2$ and $PE_3$ respectively. The C-Bus controller functions of determining the availability of destination resources are modeled by the immediate transitions $t_1, t_4, t_8, t_{11}, t_{13}$ and $t_6$. The operation of this GSPN model is exactly similar to the SPN model SPN3A. The major difference is the way the various immediate transitions are assigned probabilities of
Figure 5.4 Three Processor GSPN model - GSPN3
firing when more than one of the immediate transitions get enabled in a given marking. The switching distributions of the various immediate transitions are as follows.

There are certain immediate transitions which do not get enabled at the same time, for reasons similar to ones given in GPSN2A. We shall consider the immediate transitions connected with processor PE₁ and describe the switching distributions to them. In a similar manner, using the symmetrical nature of the model, the switching distributions of the immediate transitions associated with processors PE₂ and PE₃ can be derived.

The immediate transition t₁₉ do not get enabled at the same time with the transitions t₈ and t₁₃ due to the two conflicting timed transitions t₁₂ and t₁₇. Also t₁₉ cannot get enabled with the t₁,t₁₆ since places p₂₃ and p₂₄ cannot have tokens at the same time. Transition t₁₉ can get enabled at the same time with either t₄ or t₁₁.

The immediate transitions t₈ and t₁₃ get enabled individually or together with one of the other four immediate transitions t₁,t₄, t₁₁ and t₁₆. Thus the switching distribution for these transitions will include all these various possibilities.

The switching distributions are:

\[ t₈: \text{when } t₁₃ \text{ is enabled: } \text{Prob}=1/2; \]

when \((t₁₃,t₁)\) or \((t₁₃,t₁₄)\) or \((t₁₃,t₁₁)\) or \((t₁₃,t₁₆)\) or
(t_1, t_4) or (t_{11}, t_{16}) are enabled together: prob = 1/3; otherwise: prob = 1.

t_{13}: when t_8 is enabled: prob = 1/2; when (t_8, t_1) or (t_8, t_4) or (t_8, t_{11}) or (t_8, t_{16}) or (t_1, t_4) or (t_{11}, t_{16}) are enabled together: prob = 1/3; otherwise: prob = 1.

t_{19}: when t_4 or t_{11} is enabled: prob = 1/2; otherwise: prob = 1.

5.4 Analysis of GSPN models:

The analysis of a GSPN model, like the SPN analysis, involves first the determination of the reachability set of the GSPN which comprises of both the tangible markings and the vanishing markings (see chapter II). The second part of the analysis concerns with the determination of transition probability matrix for the Reduced Embedded Markov Chain (REM C) formed over only the tangible markings of the GSPN under study.

We have used the software package developed for this purpose in Pascal language by Marsan et al (MARSAN-85b). This software is designed to run on VAX11/780. It mainly consists of an interactive editor and various modules for determining the reachability sets, transition probability matrix and solution to the steady state probabilities of the resulting REMC. The interactive editor helps in describing the GSPN model in terms of the places, timed transitions and their firing rates, and immediate transitions along with
their switching distributions and the initial markings. The firing rates and the initial markings are treated as variable parameters so that the GSPN model can be executed for different initial markings and firing rates without specifying the model description every time. Firing rates for the timed transitions can be assigned with a fixed rate or a rate which varies with the number of tokens in its input places. Also this software allows the definition of firing rates and switching distribution with complex marking dependencies.

Determination of the reachability set:
In this the number of tangible markings and vanishing markings present in the reachability set are determined by the program GRG. The program starts with the initial marking of the GSPN model and fires all the enabled transitions in that marking. As discussed in section 2.4, the rules for the firing of the timed and immediate transitions are followed while firing the various timed and immediate transitions in a particular marking. Also in determining the nature of a new marking generated from an existing markings the rules for determining tangible and vanishing markings are used. In this software the timed transitions proceed on breadth-first manner, in which all the enabled timed transitions in a given marking are fired in sequence. The immediate transitions are fired on a depth-first manner where in the firing of successive
immediate transitions in the resulting markings are followed till a tangible marking is reached. This ordering of the markings in the reachability set help in determining the transition probabilities for the various vanishing to tangible path. These probabilities are used in determining the entries of the transition matrix defined over the tangible markings only.

Table 5.1 gives the details of reachability set for the models GSPN2A, GSPN2B, GSPN2C and GSPN3. A representative listing of the tangible and vanishing markings of a reachability set is shown in Appendix E. This is the reachability set for GSPN2B.

Determination of the steady state probabilities of the tangible markings:

The program GMD first converts the marking dependent firing definitions and the switching distributions into firing rate / probability definitions. Here a recursive descendent algorithm is employed to perform syntactic analysis of the logical definition statements and then translates them into a sequence of if-then-else statements. Then the transition probabilities for the transition matrix formed only by the tangible markings are determined by the program GMT using these definitions statements and information collected during the determination of the tangible and vanishing markings in the reachability set. This transition matrix
formed over only the tangible markings, is solved for the determination of the steady state probabilities. The GSPN software employs Gauss elimination and Gauss-Seidel iterative methods (programs GGE and GGS) for the determination of these steady state probabilities.

Determination of the values of the Performance measures:

The token density \( d_i \) for each place \( p_i \) in the given GSPN model is determined by the relation 4.3. In this context, \( y_j \) is the steady state probability of tangible marking \( j \) in the reachability set of the GSPN model.

From the various token densities, the performance measures are derived in a similar manner as done in the case of SPN analysis. As an example, the performance measures for the two processor model GSPN2B are determined by the following relations:

Processing Power \( P = d_1 + d_{13} \),

C-Bus throughput = \((1-d_9)\times \mu\),

\( PE_1 \) mean wait = \( d_1 + d_6 \),

\( PE_2 \) mean wait = \( d_{15} + d_{12} \).

5.5 Discussion of GSPN models of CUENET:

Table 5.1 shows how the reachability set of GSPN varies with the GSPN complexity of the model. It is seen that the growth of the overall reachability set is similar, but lesser than the growth of the SPN reachability set.
### Table 5.1

Details of model and reachability set size of GSPN models of CUENET.

<table>
<thead>
<tr>
<th>GSPN Model</th>
<th>No. of Places</th>
<th>No. of TMD</th>
<th>No. of TRS</th>
<th>No. of IMM</th>
<th>No. of RES</th>
<th>No. of RS</th>
<th>No. of TSt</th>
<th>No. of VSt</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSPN2A</td>
<td>15</td>
<td>6</td>
<td>6</td>
<td>70</td>
<td>40</td>
<td>30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GSPN2B</td>
<td>17</td>
<td>6</td>
<td>8</td>
<td>182</td>
<td>67</td>
<td>115</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GSPN2C</td>
<td>23</td>
<td>6</td>
<td>10</td>
<td>763</td>
<td>200</td>
<td>563</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Qu Sz = 1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(1152)*</td>
</tr>
<tr>
<td>GSPN2C</td>
<td>23</td>
<td>6</td>
<td>10</td>
<td>2828</td>
<td>612</td>
<td>2216</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Qu Sz = 2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(5832)*</td>
</tr>
<tr>
<td>GSPN3</td>
<td>25</td>
<td>12</td>
<td>11</td>
<td>826</td>
<td>479</td>
<td>347</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note:**

- **TMD TRS**: Timed Transitions.
- **IMM TRS**: Immediate Transitions.
- **R S**: Reachability Set.
- **T St**: Tangible States.
- **V St**: Vanishing States.
- **Qu Sz**: Send and receive message queue size.
- ***: Equivalent SPN reachability set size.
### Table 5.2

<table>
<thead>
<tr>
<th>MODEL</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSPN2A</td>
<td>0.830</td>
<td>1.4</td>
<td>0.4</td>
</tr>
<tr>
<td>GSPN2B</td>
<td>2.59</td>
<td>4.8</td>
<td>0.8</td>
</tr>
<tr>
<td>GSPN2C</td>
<td>24.0</td>
<td>82.9</td>
<td>2.2</td>
</tr>
<tr>
<td>(Qu Sz =1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GSPN2C</td>
<td>87.0</td>
<td>2900.0</td>
<td>25.0</td>
</tr>
<tr>
<td>(Qu Sz =2)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GSPN3</td>
<td>41.0</td>
<td>2750.0</td>
<td>10.0</td>
</tr>
</tbody>
</table>

T1: Time in secs to determine the Reachability set and form the Transition Matrix.
T2: Time in secs to determine Solution from Tr. matrix using Gaussian Elimination Method.
T3: Time in secs to determine the solution from the transition matrix using Gauss-Seidel iterative method.

Qu Sz: Send and receive message queue size.
(see Table 4.1) (This difference is quite significant for GSPN2C, for which, the figures given in parentheses correspond to the reachability set of equivalent SPN). But the states of GSPN are bifurcated into tangible and vanishing states and the computational complexity of the solution depends mainly on the size of the tangible markings of the GSPN. By observing the two tables 5.1 and 4.1 the computational complexity in the case of GSPN will be about one-third to one-half the solution complexity of SPN.

The computational time requirement, for doing GSPN analysis using the Gauss Elimination method and the Gauss-Seidel Iterative method, are shown in Table 5.2. It is observed from Table 5.2, that the Gauss elimination method takes a large amount of time, even with moderate size of the reachability (tangible) set of the GSPN. Thus for GSPNs with larger size tangible states the Gauss-Seidel iterative method was employed for obtaining the solution. The approximate solution provided by this iterative method compares quite well with those obtained using Gauss elimination method.

The results of GSPN analysis done on GSPN2A, GSPN2B, GSPN2C and GSPN3 are shown in Tables 5.3, 5.4, 5.5 and 5.6 respectively. Tables 5.3 and 5.4 gives the variation of processing power, C-Bus throughput, and processor wait time with the system load factor. In Table 5.5 the variation of processing power and C-Bus throughput with system load
<table>
<thead>
<tr>
<th>$\rho$</th>
<th>$\lambda$</th>
<th>$\mu$</th>
<th>$\text{PP}$</th>
<th>$\text{CT}$</th>
<th>$\text{P1W}$</th>
<th>$\text{P2W}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>0.4166</td>
<td>41.66</td>
<td>1.4539</td>
<td>0.2270</td>
<td>0.2731</td>
<td>0.2731</td>
</tr>
<tr>
<td>0.05</td>
<td>2.083</td>
<td>41.66</td>
<td>1.4507</td>
<td>1.1307</td>
<td>0.2747</td>
<td>0.2747</td>
</tr>
<tr>
<td>0.1</td>
<td>4.166</td>
<td>41.66</td>
<td>1.4457</td>
<td>2.2485</td>
<td>0.2772</td>
<td>0.2772</td>
</tr>
<tr>
<td>0.2</td>
<td>8.333</td>
<td>41.66</td>
<td>1.4319</td>
<td>4.4346</td>
<td>0.2821</td>
<td>0.2821</td>
</tr>
<tr>
<td>0.4</td>
<td>16.666</td>
<td>41.66</td>
<td>1.3912</td>
<td>8.5423</td>
<td>0.3044</td>
<td>0.3044</td>
</tr>
<tr>
<td>0.8</td>
<td>33.328</td>
<td>41.66</td>
<td>1.2772</td>
<td>15.4614</td>
<td>0.3614</td>
<td>0.3614</td>
</tr>
<tr>
<td>1.0</td>
<td>41.66</td>
<td>41.66</td>
<td>1.2142</td>
<td>18.2701</td>
<td>0.3929</td>
<td>0.3929</td>
</tr>
</tbody>
</table>

**NOTE:**
- PP: Processing Power
- CT: C-Bus Throughput in Messages/sec.
- P1W: wait time for PE1
- P2W: wait time for PE2
### TABLE 5.4

**GSPN ANALYSIS RESULTS OF GSPN2B:**

<table>
<thead>
<tr>
<th>(\rho)</th>
<th>(\lambda)</th>
<th>(\mu)</th>
<th>PP</th>
<th>CT</th>
<th>PLW</th>
<th>P2W</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>0.417</td>
<td>41.66</td>
<td>1.5261</td>
<td>0.2625</td>
<td>0.2369</td>
<td>0.2369</td>
</tr>
<tr>
<td>0.05</td>
<td>2.083</td>
<td>41.66</td>
<td>1.5209</td>
<td>1.3186</td>
<td>0.2395</td>
<td>0.2395</td>
</tr>
<tr>
<td>0.1</td>
<td>4.166</td>
<td>41.66</td>
<td>1.5131</td>
<td>2.6205</td>
<td>0.2434</td>
<td>0.2324</td>
</tr>
<tr>
<td>0.2</td>
<td>8.333</td>
<td>41.66</td>
<td>1.4934</td>
<td>5.1599</td>
<td>0.2533</td>
<td>0.2533</td>
</tr>
<tr>
<td>0.4</td>
<td>16.666</td>
<td>41.66</td>
<td>1.4396</td>
<td>9.8921</td>
<td>0.2802</td>
<td>0.2802</td>
</tr>
<tr>
<td>0.6</td>
<td>24.996</td>
<td>41.66</td>
<td>1.3723</td>
<td>14.0643</td>
<td>0.3139</td>
<td>0.3139</td>
</tr>
<tr>
<td>0.8</td>
<td>33.328</td>
<td>41.66</td>
<td>1.2984</td>
<td>17.6494</td>
<td>0.3508</td>
<td>0.3508</td>
</tr>
<tr>
<td>1.0</td>
<td>41.66</td>
<td>41.66</td>
<td>1.2230</td>
<td>20.6818</td>
<td>0.3885</td>
<td>0.3885</td>
</tr>
</tbody>
</table>

**NOTE:**
- PP: Processing Power.
- CT: C-Bus Throughput in messages/sec.
- PLW: wait time for PE1.
- P2W: wait time for PE2.
factor for different queue sizes are depicted.

The comparison of the results in Tables 5.3 and 5.4 with results in Tables 4.4 and 4.5 shows that the results of SPN and GSPN models for two processor CUENET agree.

In a loosely coupled system the presence of message queues at each processor facilitates a better use of the resources of the system. The comparison of the results in Tables 5.3 and 5.4 with those of Table 5.5 indicate how the increase in the C-Bus throughput has less decreasing effect on the processing power, with the introduction of message queues at each processor, in the model.

The results for three processor case (GSPN3) shown in Table 5.6 indicates that the processing power, C-Bus throughput increase proportional to the number of processors, as compared to the results of two processor case (GSPN2A). The processor wait time increases due to the increased contention for the use of C-Bus.

5.6 Aggregated GSPN model for two processor CUENET:
In chapter IV and so far in this chapter we have modeled the various message communications aspects of CUENET for two and three processors cases. In these models the action of each processor are explicitly modeled. Also we have considered the different system characteristics such as the presence of only output and input buffers and message queues at each processor.
<table>
<thead>
<tr>
<th>$\rho$</th>
<th>$\lambda$</th>
<th>$\mu$</th>
<th>QU SIZE = 1</th>
<th>QU SIZE = 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>0.417</td>
<td>41.66</td>
<td>1.5778</td>
<td>0.2876</td>
</tr>
<tr>
<td>0.05</td>
<td>2.083</td>
<td>41.66</td>
<td>1.5758</td>
<td>1.4357</td>
</tr>
<tr>
<td>0.1</td>
<td>4.166</td>
<td>41.66</td>
<td>1.5725</td>
<td>2.8647</td>
</tr>
<tr>
<td>0.2</td>
<td>8.333</td>
<td>41.66</td>
<td>1.5636</td>
<td>5.6939</td>
</tr>
<tr>
<td>0.4</td>
<td>16.666</td>
<td>41.66</td>
<td>1.5321</td>
<td>11.1404</td>
</tr>
<tr>
<td>0.6</td>
<td>24.996</td>
<td>41.66</td>
<td>1.4797</td>
<td>16.1050</td>
</tr>
<tr>
<td>0.8</td>
<td>33.328</td>
<td>41.66</td>
<td>1.4095</td>
<td>20.4126</td>
</tr>
<tr>
<td>1.0</td>
<td>41.66</td>
<td>41.66</td>
<td>1.3287</td>
<td>24.0041</td>
</tr>
</tbody>
</table>

**NOTE:**

PP : Processing Power
CT : C-Bus Throughput
in messages/sec.
TABLE 5.6

GSPN ANALYSIS RESULTS OF GSPN3:
(FOR \( \mu = 41.66 \))

<table>
<thead>
<tr>
<th>( \rho )</th>
<th>( \lambda )</th>
<th>PP</th>
<th>CT</th>
<th>P1W</th>
<th>P2W</th>
<th>P3W</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>0.417</td>
<td>2.0663</td>
<td>0.3586</td>
<td>0.3112</td>
<td>0.3112</td>
<td>0.3112</td>
</tr>
<tr>
<td>0.05</td>
<td>2.083</td>
<td>2.0603</td>
<td>1.7868</td>
<td>0.3132</td>
<td>0.3132</td>
<td>0.3132</td>
</tr>
<tr>
<td>0.1</td>
<td>4.166</td>
<td>2.0501</td>
<td>3.5526</td>
<td>0.3166</td>
<td>0.3166</td>
<td>0.3166</td>
</tr>
<tr>
<td>0.2</td>
<td>8.333</td>
<td>2.0191</td>
<td>6.9861</td>
<td>0.3270</td>
<td>0.3270</td>
<td>0.3270</td>
</tr>
<tr>
<td>0.4</td>
<td>16.666</td>
<td>1.9206</td>
<td>13.2453</td>
<td>0.3598</td>
<td>0.3598</td>
<td>0.3598</td>
</tr>
<tr>
<td>0.6</td>
<td>24.996</td>
<td>1.7919</td>
<td>18.4773</td>
<td>0.4027</td>
<td>0.4027</td>
<td>0.4027</td>
</tr>
<tr>
<td>0.8</td>
<td>33.238</td>
<td>1.6537</td>
<td>22.6734</td>
<td>0.4488</td>
<td>0.4488</td>
<td>0.4488</td>
</tr>
<tr>
<td>1.0</td>
<td>41.66</td>
<td>1.5190</td>
<td>25.9727</td>
<td>0.4937</td>
<td>0.4937</td>
<td>0.4937</td>
</tr>
</tbody>
</table>

NOTE:

PP : Processing Power.
CT : C-Bus Throughput
     in messages/sec.
P1W : wait time for PE1.
P2W : wait time for PE2.
P3W : wait time for PE3.
Table 5.1 illustrates how the model size and the reachability set size increase with increase in the details of the system being modeled. The number of places and transitions required to model two processor CUENET with message queues is almost equal to those required to model three processors without message queues. The size of the three processor model with the message queues will be 37 places and 29 transitions (12 timed and 17 immediate).

Also in GSPN models, the definition of the switching probability for the various immediate transitions can get quite involved with the increase in the number of immediate transitions in the model.

Since, in the final analysis, one is interested in the behavior of the overall system, we could represent the actions of individual processors collectively through the use of multiple tokens in selected places. By doing this the increase in the size of the model will not be high when additional details are included in the model. Further by doing such aggregation, the extension of the model for larger number of processors can be done easily with the change in the initial token distribution and/or by adding a relatively small but a new section to the model.

Since GSPN offers a better modeling capability than SPN, we have adopted GSPN for doing these aggregated CUENET models.
Fig 5.5 shows the aggregated two processor GSPN model of CUENET(GSPNR2). This model has 13 places, 8 timed transitions and 6 immediate transitions. GSPNR2 essentially models the two processor CUENET without message queues. It is similar to the GSPN2B but differs from GSPN2B in two aspects: One is we are using multiple tokens which may be used to denote the number of processors, input buffers or output buffers. The second aspect is that the rate of firing of the timed transitions vary with the number of tokens at their input places.

In GSPNR2 the activities of the various processors are modeled by places $p_1, p_2, p_3, p_4, p_5, p_7, p_9$ and timed transitions $t_1, t_2, t_{11}$ and $t_{12}$.

The C-Bus controller activities include checking the availability of the input buffer of the destination, queueing the message if necessary and the actual message transfer. The places $(p_8, p_9, p_{10}, p_{11}, p_{12})$, timed transitions $(t_6, t_7, t_8, t_{10})$ and immediate transitions $(t_3, t_4, t_5, t_9)$ model these activities.

The two immediate transitions $t_{13}$ and $t_{14}$ resolve the two deadlock situations similar to the ones in SPN2B and GSPN2B.

We shall describe the operation of the model by assuming an initial condition in which both the processors are
"active, the output and input buffers and the C-Bus controller are free. This initial state is indicated by the token distribution shown in Fig. 5.5, where place $p_1$ contains 2 tokens indicating both processors are active and places $p_3$ and $p_6$ contain 2 tokens each indicating that both the input and output buffers of the two processors are free. Place $p_5$ has one token to represent C-Bus controller being free.

The message generation by the sender and its placement in the output buffer is modeled by the timed transitions $t_1$ and $t_2$. The immediate transition $t_3$ and timed transition $t_6$ represent the C-Bus controller functions of determining the availability of the free receiver and the transfer of message to it. The message reception by the receiver processor is modeled by the timed transitions $t_{11}$ and $t_{12}$. This forms a message communication cycle.

When the destination's input buffer is busy, the C-Bus controller queues this message. This is modeled by the immediate transition $t_4$. The immediate transition $t_5$ models the queueing of second message for the same destination. The timed transitions $t_7$ and $t_8$ model the message reception by the destination processor with the additional information, one and two message are queued for the same destination. The operation of immediate transition $t_9$ is similar to $t_3$ with the information that this message was queued for this destination. Timed transition $t_{10}$ represents the C-Bus controller's message transfer of one of
the two queued messages.

In the previous models the transfer rate comprised of the message transfer rate at the sender to its output buffer, the transfer rate of the C-Bus controller and receiver processor rate of removing the message from the input buffer. In this model these components are separately modeled by different timed transitions. The timed transition \( t_2 \) fires at a marking dependent rate \( \alpha_T \), where \( \alpha_T \) is the rate at which the sender processor transfers the message to its output buffer. The timed transitions \( t_7, t_8 \) and \( t_{12} \) fire at a marking dependent rate \( \alpha_R \), where \( \alpha_R \) is the rate at which the receiver processor removes the message from its input buffer. The timed transitions \( t_6 \) and \( t_{10} \) fire at a constant rate \( \mu_C \), where \( \mu_C \) is the rate at which the C-bus controller transfers messages from the senders' output buffers to the receivers' input buffers. The message transfer rate \( \mu \) used in the earlier models and these three rates are related by the following expression:

\[
\frac{1}{\mu} = \frac{1}{\mu_C} + \frac{1}{\alpha_T} + \frac{1}{\alpha_R} \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad (5.1)
\]

The immediate transitions \( t_3, t_4 \) and \( t_5 \) model the selection of the receiver's input buffer. At certain markings more than one i.e either \( t_3, t_4 \) or \( t_3, t_5 \) will be enabled indicating that the message may be directed to a free receiver's input buffer or can queue for a busy input buffer. To model this we have to assign probabilities of firing to these transitions, when they are enabled together.
This probability will depend on the marking in which this immediate transition gets enabled. Thus the switching distribution for each immediate transition in terms of the tokens at their input places can be determined. The switching distribution for the transitions $t_3, t_4, t_5$ and $t_9$ are as given below:

For $t_3$: $m_6/\text{PROCS}$,

For $t_4$: $m_7/\text{PROCS}$,

For $t_5$: $m_9/\text{PROCS}$,

For $t_9$: when $t_3$ is enabled, $m_{10}/\text{PROCS}$

Otherwise 1.

where PROCS is the number of processors in the model.

The above switching distributions holds good for all marking in the reachability set of the GSPN.

Fig 5.6 shows the reduced GSPN model including the send and message queues(GSPNRQ2). This GSPN model has 17 places, 8 timed transitions and 8 immediate transitions. The operation of this model is essentially same as the GSPNR2. The extra information contained is the queueing of messages at the sending and receiving ends. This aspect is modeled by the immediate transitions $t_2$ and $t_{14}$. The immediate transition $t_2$ models the message queueing at the senders' side and $t_{14}$ models the reception of the messages contained in the receive message queue. Thus the message generation and placing the generated message in the sender processor send queue are modeled by the timed transition $t_1$ and
Figure 5.6 Two processor (with message queues)

Aggregated GSPN model - GSPNRQ2
immediate transition \( t_2 \). Timed transition \( t_3 \) models the movement of the message from the send queue to the output buffer of the sender processor.

For the sake of simplicity, in this model we have assumed that the interaction of the processor is minimal during the movement of the message from the send queue to the output buffer. Alternately, this model also represents the situation if a coprocessor, like the 82586 LAN communication controller [INTEL-84], were to be present to take care of the message communication functions of the processor.

The immediate transitions \( t_4, t_5 \) and \( t_6 \) model the selection of free or busy destination receiver for message transfer. Timed transition \( t_7 \) models the message transmission by the C-Bus controller to a free receiver's input buffer. The timed transition \( t_8 \) represents the movement of message from the receiver's input buffer to its receive queue.

The timed transition \( t_{13} \) and immediate transition \( t_{14} \) together model the reception of a message in the receive queue by a processor.

The timed transitions \( t_9 \) and \( t_{10} \) model the same operation as done by the timed transition \( t_8 \) with the additional information that one and two output buffers are queued, respectively, for that busy receiver's input buffer.
The timed transition $t_{12}$ represents the transfer of a message from one of the two queued output buffers to that receiver's input buffer.

The immediate transition $t_{11}$ models the selection of the queued output buffer message for transfer by the C-Bus controller to the required receiver's input buffer.

The immediate transitions $t_{15}$ and $t_{16}$ resolve the deadlock situations similar to $t_{13}$ and $t_{14}$ in GSPNR2.

The firing rates of the timed transitions are similar to that in GSPNR2. The timed transitions $t_1$ and $t_{13}$ fire at a marking dependent rate $m_1X\lambda/2$, the timed transitions $t_7$ and $t_{12}$ fire at a constant rate $\mu_C$ and the transition $t_3$ fires at the marking dependent rate $m_5X\alpha_T$. Transitions $t_8,t_9$ and $t_{10}$ fire at a marking dependent rates $m_{10}X\alpha_R, m_{13}X\alpha_R$ and $m_{15}X\alpha_R$ respectively. The rates $\mu_C, \alpha_T, \alpha_R$ have the same relation as given in equation 5.1. As it happens in GSPNR2, in this model also in certain markings more than one immediate transition can get enabled at the same time. The immediate transitions $t_2,t_4,t_5$ and $t_6$ will get enabled singly or in groups of two or three. Also the immediate transitions $t_{11}$ and $t_{14}$ get enabled simultaneously in certain markings.

In these groups, for the sake of simplicity, we assume that the immediate transition modeling the processor activity will get priority. For example, if immediate
transitions \( t_2 \) and \( t_6 \) get enabled in a given marking then a probability of 1 is assigned to the firing of \( t_2 \) while a probability of 0 is assigned to \( t_6 \). With this assumptions the switching distributions for the various immediate transitions are as follows:

\( t_2 \) and \( t_{14} \) will fire with probability 1,

\( t_4 \) : when \( t_2 \) enabled : probability = 0,
otherwise probability = \( m_8/\text{PROCS} \),

\( t_5 \) : when \( t_2 \) is enabled : probability = 0,
otherwise, probability = \( m_{10}/\text{PROCS} \),

\( t_6 \) : when \( t_2 \) enabled : probability = 0,
otherwise, probability = \( m_{13}/\text{PROCS} \),

\( t_{11} \) : when \( t_{14} \) enabled : probability = 0,
otherwise, probability = 1,

where \( m_i \) is the number of tokens in place \( p_i \) and \( \text{PROCS} \) = number of processors in the model.

The extension of the reduced two processor GSPN models (GSPNR2,GSPNRQ2) to three and higher number of processors is quite straightforward. Fig. 5.7 shows the three processor version of GSPNRQ. Here the processors activities portion do not get altered, only the number of tokens in the initial marking gets changed. The portion modeling the C-Bus controller needs to be extended to queue one more processor. In Fig. 5.7 the places \( p_{18}, p_{19}, p_{20} \), the timed transitions \( t_{18}, t_{20} \) and the immediate transitions \( t_{17}, t_{19} \) represent this extension of C-Bus queue (shown enclosed in dotted lines).
Thus this three processor aggregated GSPN model has 20 places, 10 timed and 10 immediate transitions. To model a system with "i" processors this portion of the C-Bus controller queues must be extended \((i-2)\) times. Thus for modeling a 5 processor CUENET we need three such extra portions and thus the total number of places will come to 26, timed transitions 14 and immediate transitions also to 14.

5.7 Discussion of Aggregated GSPN models of CUENET:

Table 5.7 gives the details regarding the size of the aggregated GSPN models and the size of their reachability set. By comparing the entries in Table 5.7 with those in Table 5.1 we observe that the aggregation has resulted in smaller and compact GSPN models for the two processor CUENET. The savings one obtains in terms of places and transitions is quite significant as one progresses from two processor to three processor aggregate model with message queues.

But the overall reachability set for both the aggregate and detailed models are of the same magnitude. This is due to the fact that with the aggregate model we are still presenting the same or greater amount of details of the system. Hence the reason for the same size in the reachability sets.
### TABLE 5.7

DETAILS OF MODEL AND REACHABILITY SET SIZE FOR AGGREGATE GSPN MODELS:

<table>
<thead>
<tr>
<th>GSPN MODEL</th>
<th>NO. OF PLACES</th>
<th>NO. OF TMD TRS</th>
<th>NO. OF IMM TRS</th>
<th>NO. OF SIZE OF R S</th>
<th>NO. OF T St</th>
<th>NO. OF V St</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSPNR2</td>
<td>13</td>
<td>8</td>
<td>6</td>
<td>129</td>
<td>90</td>
<td>39</td>
</tr>
<tr>
<td>GSPNRQ2</td>
<td>17</td>
<td>8</td>
<td>8</td>
<td>876</td>
<td>342</td>
<td>534</td>
</tr>
<tr>
<td>(Qu Sz=1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GSPNRQ2</td>
<td>17</td>
<td>8</td>
<td>8</td>
<td>2012</td>
<td>708</td>
<td>2012</td>
</tr>
<tr>
<td>(Qu Sz=2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GSPNRQ2</td>
<td>17</td>
<td>8</td>
<td>8</td>
<td>3616</td>
<td>1202</td>
<td>2414</td>
</tr>
<tr>
<td>(Qu Sz=3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOMP</td>
<td>9</td>
<td>6</td>
<td>5</td>
<td>24</td>
<td>16</td>
<td>8</td>
</tr>
</tbody>
</table>

**NOTE:**
- TMD TRS: Timed Transitions.
- IMM TRS: Immediate Transitions.
- R S: Reachability Set.
- T St: Tangible States.
- V St: Vanishing States.
- Qu Sz: Send and Receive message queue size.
<table>
<thead>
<tr>
<th>MODEL</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSPNR2</td>
<td>2.89</td>
<td>2.40</td>
<td>0.5</td>
</tr>
<tr>
<td>GSPNRQ2</td>
<td>22.74</td>
<td>2716</td>
<td>8.5</td>
</tr>
<tr>
<td>(Qu Sz=1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GSPNRQ2</td>
<td>55.0</td>
<td>6000</td>
<td>*</td>
</tr>
<tr>
<td>(Qu Sz=2)</td>
<td></td>
<td></td>
<td>20.0</td>
</tr>
<tr>
<td>GSPNRQ2</td>
<td>96.0</td>
<td>9000</td>
<td>*</td>
</tr>
<tr>
<td>(Qu Sz=3)</td>
<td></td>
<td></td>
<td>100.0</td>
</tr>
</tbody>
</table>

**NOTE:**

- **T1**: Time in secs to determine the reachability set and form transition matrix.
- **T2**: Time in secs to determine the solution from the transition matrix by Gaussian elimination.
- **T3**: Time in secs to determine the solution from the transition matrix by Gauss-Seidel iterative method.
- ***: Estimated value.
- **Qu Sz**: Send and receive message queue size.
The last entry in Table 5.7 corresponds to the reachability set of the reduced GSPN model of TOMP multiprocessor, modeling 5 processors, 3 common memories and 2 buses. Here again, as seen in SPN analysis, the loose coupled nature of CUENET results in a larger size reachability sets.

Table 5.8 gives the computational time required for doing GSPN analysis of these aggregate models.

From these results, we see that the aggregate GSPN model does not provide much computational advantage. But it does provide the advantage of compactness and ease of expansion, as illustrated in the previous section, to represent the system with higher number of processors.

Table 5.9 gives the GSPN analysis results of GSPNR2, while tables 5.10 and 5.11 contain the results of GSPNRQ2. The Table 5.9 gives the GSPN analysis results of GSPNR2, which is the aggregate version of the two processor model GSPN2B. Here the different aspects of message communication are modeled in more detail as compared to GSPN2B. The size of average inbuffer queue length is due to the fact in this model, the situation of processor sending a message to itself is also modeled. When a processor is sending a message to itself, the other processors will have to wait for an extra amount of time, which is reflected by the average size of the inbuffer queue.
TABLE 5.9
GSPN ANALYSIS RESULTS OF GSPNR2:

<table>
<thead>
<tr>
<th>$\rho$</th>
<th>$\lambda$</th>
<th>$\mu$</th>
<th>PP</th>
<th>CT</th>
<th>CBQ</th>
<th>IBFQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>0.417</td>
<td>41.66</td>
<td>1.6887</td>
<td>0.2364</td>
<td>0.0000</td>
<td>0.5618</td>
</tr>
<tr>
<td>0.05</td>
<td>2.083</td>
<td>41.66</td>
<td>1.6662</td>
<td>1.1684</td>
<td>0.0000</td>
<td>0.5568</td>
</tr>
<tr>
<td>0.1</td>
<td>4.166</td>
<td>41.66</td>
<td>1.6386</td>
<td>2.3034</td>
<td>0.0002</td>
<td>0.5506</td>
</tr>
<tr>
<td>0.2</td>
<td>8.332</td>
<td>41.66</td>
<td>1.5848</td>
<td>4.4751</td>
<td>0.0001</td>
<td>0.5390</td>
</tr>
<tr>
<td>0.4</td>
<td>16.664</td>
<td>41.66</td>
<td>1.4837</td>
<td>8.4406</td>
<td>0.0003</td>
<td>0.5181</td>
</tr>
<tr>
<td>0.6</td>
<td>24.996</td>
<td>41.66</td>
<td>1.3914</td>
<td>11.9441</td>
<td>0.0007</td>
<td>0.4999</td>
</tr>
<tr>
<td>0.8</td>
<td>33.328</td>
<td>41.66</td>
<td>1.3077</td>
<td>15.0424</td>
<td>0.0012</td>
<td>0.4840</td>
</tr>
<tr>
<td>1.0</td>
<td>41.66</td>
<td>41.66</td>
<td>1.2317</td>
<td>17.7877</td>
<td>0.0016</td>
<td>0.4700</td>
</tr>
</tbody>
</table>

NOTE:
- PP : Processing Power.
- CT : C-Bus Throughput
- CBQ : C-Bus average queue length.
- IBFQ : Input Buffers average queue length
### TABLE 5.10

GSPN ANALYSIS RESULTS OF GSPNRQ2:
(Processing Power for different queue sizes.)

<table>
<thead>
<tr>
<th>$\rho$</th>
<th>$\lambda$</th>
<th>$\mu$</th>
<th>PP1</th>
<th>PP2</th>
<th>PP3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>0.417</td>
<td>41.66</td>
<td>1.8093</td>
<td>1.8617</td>
<td>1.8916</td>
</tr>
<tr>
<td>0.05</td>
<td>2.083</td>
<td>41.66</td>
<td>1.8062</td>
<td>1.8594</td>
<td>1.8898</td>
</tr>
<tr>
<td>0.1</td>
<td>4.166</td>
<td>41.66</td>
<td>1.8017</td>
<td>1.8562</td>
<td>1.8872</td>
</tr>
<tr>
<td>0.2</td>
<td>8.332</td>
<td>41.66</td>
<td>1.7908</td>
<td>1.8484</td>
<td>1.8810</td>
</tr>
<tr>
<td>0.4</td>
<td>16.664</td>
<td>41.66</td>
<td>1.7610</td>
<td>1.8279</td>
<td>1.8647</td>
</tr>
<tr>
<td>0.6</td>
<td>24.996</td>
<td>41.66</td>
<td>1.7216</td>
<td>1.8013</td>
<td>1.8436</td>
</tr>
<tr>
<td>0.8</td>
<td>33.328</td>
<td>41.66</td>
<td>1.6739</td>
<td>1.7690</td>
<td>1.8180</td>
</tr>
<tr>
<td>1.0</td>
<td>41.66</td>
<td>41.66</td>
<td>1.6200</td>
<td>1.7311</td>
<td>1.7881</td>
</tr>
</tbody>
</table>

**NOTE:**

PP1 : Processing Power for
Queue size = 1 message.

PP2 : Processing Power for
Queue size = 2 messages.

PP3 : Processing Power for
Queue size = 3 messages.
TABLE 5.11

GSPN ANALYSIS RESULTS OF GSPRQ2:
(C-Bus Throughput for different queue sizes.)

<table>
<thead>
<tr>
<th>$\rho$</th>
<th>$\lambda$</th>
<th>$\mu$</th>
<th>CT1</th>
<th>CT2</th>
<th>CT3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>0.417</td>
<td>41.66</td>
<td>0.2664</td>
<td>0.2793</td>
<td>0.2867</td>
</tr>
<tr>
<td>0.05</td>
<td>2.083</td>
<td>41.66</td>
<td>1.3377</td>
<td>1.4045</td>
<td>1.4424</td>
</tr>
<tr>
<td>0.1</td>
<td>4.166</td>
<td>41.66</td>
<td>2.6874</td>
<td>2.8270</td>
<td>2.9062</td>
</tr>
<tr>
<td>0.2</td>
<td>8.332</td>
<td>41.66</td>
<td>5.4121</td>
<td>5.7171</td>
<td>5.8894</td>
</tr>
<tr>
<td>0.4</td>
<td>16.664</td>
<td>41.66</td>
<td>10.8797</td>
<td>11.6106</td>
<td>12.0159</td>
</tr>
<tr>
<td>0.6</td>
<td>24.996</td>
<td>41.66</td>
<td>16.2352</td>
<td>17.5440</td>
<td>18.2524</td>
</tr>
<tr>
<td>0.8</td>
<td>33.328</td>
<td>41.66</td>
<td>21.3500</td>
<td>23.4014</td>
<td>24.4940</td>
</tr>
<tr>
<td>1.0</td>
<td>41.66</td>
<td>41.66</td>
<td>26.1281</td>
<td>29.0713</td>
<td>30.6399</td>
</tr>
</tbody>
</table>

NOTE:
- **CT1**: C-Bus Throughput in messages/sec for queue size = 1 message.
- **CT2**: C-Bus Throughput in messages/sec for queue size = 2 messages.
- **CT3**: C-Bus Throughput in messages/sec for queue size = 3 messages.
The improvement in the processing power and C-Bus throughput with the inclusion of message queues at the send and receive points is indicated in the results presented in Tables 5.10 and 5.11 respectively.

The study of these detailed and aggregate GSPN models of CUENET suggests that the GSPN modeling technique can be utilized to study a small loosely coupled DCS, say with two or three processors.
CHAPTER VI
CONCLUSIONS.

This thesis aims at the performance evaluation of CUENET (Concordia University Educational Network), a loosely coupled Distributed Computing System (DCS), using Stochastic Petrinets (SPNs) and Generalized Stochastic Petrinets (GSPNs).

Of late, SPNs and GSPNs are employed in the performance studies of tightly coupled DCS like the Torino MultiProcessor (TOMP) [MARSAN-83], and the Fault Tolerant MultiProcessors (FTMP) [WOOD-84]. These recent applications of SPNs and GSPNs on one hand and the presence of CUENET here in our department on the other hand are the main motivating factors in undertaking this thesis work. Also, to the author's knowledge, to date, these techniques have not been employed in the performance study of a loosely coupled DCS.

The SPN models, presented in chapter IV, for the two and three processor CUENET reveal two aspects:

1. The ease with which concurrency and conflicts in the system can be modeled. Also, using the Petrinet model properties like boundedness and liveness, the modeled system can be easily checked for deadlock.

2. The reachability set of the SPN grows enormously as one proceeds to add more and more details of the system into the
model. Table 4.1 illustrates this increase.

The large size of the reachability set reported in this thesis is attributable to the loose coupling and the protocol details employed in a network such as CUENET. The autonomous and asynchronous nature of a loosely coupled system results in a large number of system states. This contrasts with the size of the SPNs and reachability sets of the tightly coupled TOMP shown in Table 4.3. The large size of the reachability set results in high computational times to obtain the steady state solution of the underlying discrete Markov chain.

The classification of transitions into timed and immediate 'types' in GSPN allows one to express more details of the system for the same time complexity of the model. The GSPN models of chapter V illustrate this aspect. Table 5.1 gives the details of the reachability sets of the GSPN models. The models GSPN2A, GSPN2B have lesser number of states as compared to their SPN counterparts, where the level of details are the same. The GSPN2C models the two processor CUENET with additional details of message queueing at each processor. This has far lesser number of states as compared to an equivalent SPN. Also the results of the two processor GSPN models GSPN2A and GSPN2B compare well with the results of the corresponding SPN models SPN2A and SPN2B, as indicated by the Tables 5.3, 5.4 and Tables 4.4, 4.5 respectively.
The main advantage of aggregate models is in their compactness and the ease of expansion of the model to represent additional processors. However, in an aggregate model one cannot get the details of the activities of individual processors. The aggregate GSPN models for two processor CUENET are shown in figures 5.5, 5.6; and for three processor CUENET is shown in fig 5.7. Table 5.7 gives the details of the reachability sets of the aggregate models. Here also we see the fast growth of the reachability set with addition of more system details in the model.

For the analysis of GSPN models, the software package supplied by Dr. M.A.Marsan of Politecnico de Torino, Torino, Italy [MARSAN-85b] was used. This software was run on VAX 11/780. The SPN models were analysed by means of a set of programs developed by the author on Cyber 835. The computational times for GSPN models were given in tables 5.2, 5.8 and for SPN models in table 4.2. From these results it is observed that analysis of loosely coupled systems, even with as few as two processors, incurs a substantial amount of increase in the computational times with addition of more details of the system. In the GSPN analysis of the two processor model, the computation time varies from 0.5 secs to 100 secs with the addition of message queues at each processor. Hence the analysis of tens and hundreds of processors would be impractical in this way.
For the analysis and modeling of loosely coupled systems involving hundreds of processors, like the Cosmic Cube [SEITZ-85], methodologies of applying GSPN models have to be evolved in future. It is an open problem worth future study. One way to study them could be to decompose a large system into a set of small and computationally manageable sizes and solve this set, perhaps using a DCS.
REFERENCES


**[ENSLOW-78]** Philip H. Enslow Jr. - What is a "Distributed Data processing system?" - IEEE Computer, January 1978, P3-11.


**[GROSS-82]** Clifford Grossner - The design and implementation of CUENET, a reconfigurable network of loosely coupled Microcomputers - M. Comp. Sc Thesis, 1982, Concordia University, Montreal, Canada.

**[GUPTA-84]** Amar Gupta and Hoo-min D. Toong - The first decade of personal computers - IEEE Proceedings, Vol.72, No.3, March 1984, P246-258


**[HURA-81]** G.S. Hura et al - State equation representation of logic operations through Petri-nets - Proceedings of IEEE, Vol.69, No.2, April 81, P485-487.


University of California, Los Angeles, 1981.


APPENDIX A:

Program Listing for Determining the SPN reachability set using Prime number method.
PROGRAM REACPET ( INPUT, OUTPUT )
C THIS PROGRAM TAKES THE DESCRIPTION OF A STOCHASTIC PETRINET AND DERIVES
C THE REACHABILITY SET FOR IT.
C
REAL PROB(200)
DIMENSION BTRANS(50), DENS(15), NPRIME(50), NSTATE(2000), NINPUT(50)
DIMENSION NTEMP(50), NOUT(50)
DATA (NPRIME(I), I=1,50) /2,3,5,7,11,13,17,19,23,29,31,
 37,41,43,47,53,59,61,67,71,
 73,79,83,89,97,101,103,107,109,113,127,131,137,139,
149,151,157,163,167,173,179,181,191,193,197,199,211,
223,227,229/
PRINT 10
10 FORMAT ('1', ' THE FOLLOWING IS THE DESCRIPTION OF THE',
   2 ' GIVEN PETRINET.' )
READ 11,NPLACE
11 FORMAT ( I8 )
PRINT 15,NPLACE
15 FORMAT ('0', ' THE NUMBER OF PLACES FOR THE GIVEN',
   2 ' PETRINET IS ', I5 )
READ 11,NTRANS
PRINT 20,NTRANS
20 FORMAT ('0', ' THE NUMBER OF TRANSITIONS FOR THE GIVEN',
   2 ' PETRINET IS ', I5 )

C LOOP THROUGH THE TRANSITIONS CREATING THE INPUT AND OUTPUT
C
PRINT 22
22 FORMAT ('1', ' FOLLOWING IS THE DESCRIPTION OF THE INPUT AND',
   2 ' OUTPUT PLACES FOR EACH TRANSITION IN THE NET.' )
   0 50, 1=1,NTRANS
NINPUT(I)=1
NOUT(I)=1
25 READ 11,NP
PRINT 30,I,NP
30 FORMAT ('0', ' THE INPUT PLACE FOR THE TRANSITION ', IS, ', ', IS, ', IS')
   IF (NP.GT.NPLACE) GO TO 25
   IF (NP.LE.0) GO TO 35
NINPUT(I) = NINPUT(I) + NPRIME(NP)
GO TO 25
35 READ 11,NP
   PRINT 40,I,NP
40 FORMAT ('0', ' THE OUTPUT PLACE FOR TRANSITION ', IS, ', ', IS, ', IS')
   IF (NP.GT.NPLACE) GO TO 35
   IF (NP.LE.0) GO TO 50
NOUT(I) = NOUT(I) + NPRIME(NP)
GO TO 35
50 CONTINUE
PRINT 23
23 FORMAT ('1', ' THE AVERAGE FIRING RATES FOR THE TRANSITION ARE: ',/)
DO 54 I=1,NTRANS
READ 12, RTRANS(I)
FORMAT(F8.4)
PRINT 53, I, RTRANS(I)
FORMAT ('0', 'THE AVERAGE TRANSITION RATE OF TRANSITION', I4, 2 ' IS ', F10.4 )
CONTINUE
PRINT 55
FORMAT('0', 'END OF THE DESCRIPTION OF THE PETRINET. ')
PRINT 60
FORMAT ('1', 'DESCRIPTION OF THE INITIAL MARKING OF THE NET. ')
NSTATE(1) = 1
DO 65, I = 1, NPLACE
READ 11, NTOKEN
PRINT 61, I, NTOKEN
NSTATE(1) = NSTATE(1)*NPRIME(I)*NTOKEN
CONTINUE
NOW GENERATE THE VARIOUS REACHABLE STATES.
C
NUMST = 1
NEWFLG = 0
LIMIT = NUMST
DO 80, M = 1, LIMIT
TERMNODE = 0
DO 70, J = 1, NTRANS
NTEMP(J) = 0
IF (MOD(NSTATE(M), NINPUT(J)) .NE. 0) GO TO 70
TERMNODE = 1
NTEMP(J) = NSTATE(M)/NINPUT(J)*NOUT(J)
CONTINUE
IF (TERMNODE .NE. 0) GO TO 71
PRINT 72, M
2 FORMAT('0', 'THE MARKING WITH STATE', I3, ' IS A', 2 ' TERMINAL MARKING. ')
C
C PUT THE NEW STATE IN WITH THE OLD IN ORDER.
C
71 DO 80, J = 1, NTRANS
IF (NTEMP(J).EQ.0) GO TO 80
DO 75, I = 1, NUMST
IF (NTEMP(J).EQ.NSTATE(I)) GO TO 80
CONTINUE
NUMST = NUMST + 1
NSTATE(NUMST) = NTEMP(J)
NEWFLG = 1
CONTINUE
C
C SEE IF ANY NEW STATES.
C
IF (NEWFLG .NE. 0) GO TO 67
PRINT 85, NUMST
FORMAT('0', 'THERE ARE ', I5, ' STATES IN THE REACHABILITY', 2 ' SET FOR THE GIVEN PETRINET. ')
2
PRINT 95
95    FORMAT('1', 'THE MARKINGS CORRESPONDING TO THE VARIOUS 
   3   STATES IN REACHABILITY SET OF THE GIVEN 
   4   PETRI NET ARE AS SHOWN.');
   DO 110 I = 1, NUMST
   DO 115 J = 1, NPLACE
      NTEMP(J) = 0
      NUMB = NSTATE(I)
   120   IF (MOD(NUMB, NPRIME(J)) .NE. 0) GO TO 115
   C   C COUNT THE TOKENS AND REMOVE THEM.
   C
      NTEMP(J) = NTEMP(J) + 1
      NUMB = NUMB / NPRIME(J)
   GO TO 120
   115  CONTINUE
   PRINT 125, I, (NTEMP(J), J = 1, NPLACE)
   125  FORMAT ('0', 'STATE 13', ':', '40I3')
   110  CONTINUE
   STOP
   END
APPENDIX B:

Program Listing for determining the SPN reachability set using Binary number method.
PROGRAM BIREPET (INPUT, OUTPUT)
REAL PROB(200)
DIMENSION NSTATE (6000), NINPUT(100), NOUT(100), NTEMP(1000),
1 RTRANS(100)
STATIME = SECOND()
PRINT 80
10 FORMAT ('1', ' THE FOLLOWING IS THE DESCRIPTION OF THE ',
2 ' GIVEN PETRINET. ')
READ 15, NPLACE
15 FORMAT (18)
PRINT 20, NPLACE
20 FORMAT ('0', ' THE NUMBER OF PLACES FOR THE GIVEN PETRINET IS ', I5)
READ 15, NTRANS
PRINT 25, NTRANS
25 FORMAT ('0', ' THE NUMBER OF TRANSITIONS FOR THE GIVEN ',
2 ' PETRINET IS ', I5)
PRINT 30
30 FORMAT ('1', ' FOLLOWING IS THE DESCRIPTION OF THE INPUT AND ',
3 ' OUTPUT PLACES FOR EACH TRANSITION IN THE PETRINET. ')
DO 35, I = 1, NTRANS
   NINPUT(I) = 0
   NOUT(I) = 0
40 READ 15, NP
PRINT 45, I, NP
45 FORMAT ('0', ' THE INPUT PLACE FOR TRANSITION ', I5,
4 ' IS ', I5)
   IF(NP LE 0) GO TO 50
   IF(NP GT NPLACE) GO TO 40
   NINPUT(I) = NINPUT(I) + 2**NP - 1)
   GO TO 40
50 READ 15, NP
PRINT 55, I, NP
55 FORMAT ('0', ' THE OUTPUT PLACE FOR TRANSITION ', I5,
5 ' IS ', I5)
   IF(NP LE 0) GO TO 35
   IF(NP GT NPLACE) GO TO 50
   NOUT(I) = NOUT(I) + 2**NP - 1)
   GO TO 50
35 CONTINUE
DO 51, I = 1, NTRANS
PRINT 60, I, NINPUT(I), NOUT(I)
60 FORMAT ('0', ' FOR TRANSITION ', I5, ' INPUT = ', I20,
4 ' OUTPUT = ', I20)
51 CONTINUE
PRINT 66
36 FORMAT ('1', ' THE AVERAGE FIRING RATES FOR THE TRANSITIONS ARE: '
37 DO 37, I = 1, NTRANS
READ 16, RTRANS(I)
16 FORMAT (F8.4)
PRINT 38, I, RTRANS(I)
38 FORMAT ('0', ' THE AVERAGE FIRING RATE OF TRANSITION ',
39 ' IS ', F19.5)
37 CONTINUE
PRINT 65
FORMAT('0', ' END OF DESCRIPTION OF THE GIVEN PERNET. ')
PRINT 70
FORMAT('1', ' DESCRIPTION OF THE INITIAL MARKING OF THE PERNET. ')
NLSTATE(1) = 0
DO 75, I = 1, NPLACE
READ 15,NTOKEN
PRINT 80, I,NTOKEN
FORMAT('0', ' THE PLACE ',I5,' HAS ',I4,' TOKENS IN IT. ')
NLSTATE(I) = NLSTATE(I) + NTOKEN*(2**(I-1))
PRINT 76,NLSTATE(I)
75 CONTINUE
C NOW GENERATE THE VARIOUS REACHABLE STATES.
C
NUMST = 1
NEWFLG = 0
IF(NUMST.EQ.1) ISTART = 1
LIMIT = NUMST
PRINT 86,LIMIT
FORMAT('0', ' LIMIT = ',I5)
DO 90, M= ISTART,LIMIT
TERNODE = 0
DO 95, J= 1, NTRANS
NTEMP(J) = 0
IAND = NLSTATE(M).AND.NINPUT(J)
INEQ = IAND.NEQV.NINPUT(J)
IF(INEQ.NE.0) GO TO 95
TERNODE = 1
INTRES = NLSTATE(M).AND.(.NOT.NINPUT(J))
NTEMP(J) = INTRES.OR.NOUT(J)
95 CONTINUE
IF(TERNODE.NE.0)GO TO 100
PRINT 105,M,NLSTATE(M)
FORMAT('0', ' THE MARKING WITH STATE ',I4,' IS A ')
5 ' TERMINAL MARKING. AND NSTATE(M) = ',I10
100 DO 110,J= 1,NTRANS
IF(NTEMP(J).EQ.0) GO TO 110
DO 115, I= 1,NUMST
IF(NTEMP(J).EQ.NLSTATE(I)) GO TO 110
115 CONTINUE
NUMST = NUMST + 1
NLSTATE(NUMST)**2 = NTEMP(J)
NEWFLG = 1
110 CONTINUE
90 CONTINUE
C SEE IF ANY NEW STATES.
C
IF(NEWFLG.NE.0) THEN
ISTART = LIMIT
PRINT 300,ISTART
300 FORMAT('0', ' ISTART= ',I4)
GO TO 85
ENDIF
PRINT 120, NUMST
120 FORMAT('I1,', ' THERE ARE ', I5, ' STATES IN THE REACHABILITY ',
4 ' SET OF THE GIVEN PETRINET. ')
DO 125, I = 1, NUMST
PRINT 130, I, NSTATE(I)
130 FORMAT('0', ' THE STATE NUMBER OF THE MARKING ', I5,
5 ' IS ', 120)
125 CONTINUE
C DETERMINE THE TOKENS IN VARIOUS PLACES FOR EACH STATE AND PRINT THEM...
C
DO 140, I = 1, NUMST
NUMB = NSTATE(I)
DO 145, J = 1, NPLACE
NTEMP(J) = 0
IF(NUMB.EQ.1) GO TO 155
IF(NUMB.EQ.0) GO TO 145
IODD = MOD(NUMB, 2)
IF(IODD.EQ.1) GO TO 155
NUMB = NUMB/2
GO TO 145
155 NUMB = NUMB - 1
NUMB = NUMB/2
NTEMP(J) = NTEMP(J) + 1
145 CONTINUE
PRINT 160, I, (NTEMP(J), J = 1, NPLACE)
160 FORMAT('0', ' STATE ', I3, ': ', 4013)
140 CONTINUE
ONETIME = SECOND()
THEINT = ONETIME - STATIME
PRINT 500, THEINT
500 FORMAT('0', ' THE TIME FOR DETERMINING THE REACHABILITY ',
2 ' SET IS: ', F15.6)
STOP
END
APPENDIX C:

Program Listing for determining the GSPN reachability set using Hexadecimal number method.
PROGRAM BIREPET (INPUT, OUTPUT)
DIMENSION NSTATE(4000, 3), NIMITM(50), NOUTIM(50), NTEMP(50, 3),
1 NTRANS(50), NTXMTP(50), NINMM(50), NOUTIM(50),
4 NTMANG(4000, 3), NSTVAN(4000, 3)

STATEM = SECOND()
PRINT 10
10 FORMAT('1', ' THE FOLLOWING IS THE DESCRIPTION OF THE',
2 ' GIVEN GENERALIZED STOCHASTIC PETRINET. ')
READ 15, NPLACE
15 FORMAT(18) *
PRINT 20, NPLACE
20 FORMAT('0', ' THE NUMBER OF PLACES FOR THE GIVEN PETRINET IS ', I5)
READ 15, NTIMTR
PRINT 25, NTIMTR
25 FORMAT('0', ' THE NUMBER OF TIMED TRANSITIONS FOR THE ',
2 ' GIVEN GENERALIZED STOCHASTIC PETRINET IS ', I5)
READ 15, NIMMTR
PRINT 26, NIMMTR
26 FORMAT('0', ' THE NUMBER OF IMMEDIATE TRANSITIONS IN THE ',
2 ' GIVEN GENERALIZED STOCHASTIC PETRINET IS ', I5)
PRINT 30
30 FORMAT('1', ' FOLLOWING IS THE DESCRIPTION OF THE INPUT AND',
3 ' OUTPUT PLACES FOR EACH TIMED TRANSITION IN THE ',
4 ' GENERALIZED STOCHASTIC PETRINET. ')
DO 35, I = 1, NTIMTR
NIMITM(I) = 0
NOUTIM(I) = 0
40 READ 15, NP
PRINT 45, I, NP
45 FORMAT('0', ' THE INPUT PLACE FOR TIMED TRANSITION ', I5,
4 ' IS ', I5)
IF(NP.LE.0) GO TO 50
IF(NP.GT.NPLACE) GO TO 40
NIMITM(I) = NIMITM(I) + 2**(NP-1)
GO TO 40
50 READ 15, NP
PRINT 55, I, NP
55 FORMAT('0', ' THE OUTPUT PLACE FOR TIMED TRANSITION ', I5,
4 ' IS ', I5)
IF(NP.LE.0) GO TO 35
IF(NP.GT.NPLACE) GO TO 50
NOUTIM(I) = NOUTIM(I) + 2**(NP-1)
GO TO 50
35 CONTINUE.
PRINT 31
31 FORMAT('0', ' THE FOLLOWING IS THE INPUT AND OUTPUT PLACE ',
2 ' DESCRIPTION FOR EACH IMMEDIATE TRANSITIONS OF THE GIVEN ',
3 ' GENERALISED STOCHASTIC PETRINET. ')
DO 300, I = 1, NIMMTR
NINMM(I) = 0
NOUTMM(I) = 0
305 READ 15, NP
PRINT 310, I, NP
310 FORMAT('0', ' THE INPUT PLACE FOR IMMEDIATE TRANSITION ', I5,
4 ' IS ',I5)
IF(NP.LE.0) GO TO 315
IF(NP.GT.NPLACE) GO TO 305
NINMM(I) = NINMM(I) + 2**(NP-1)
GO TO 305
315 READ 15,NP
PRINT 320,1,NP
320 FORMAT(‘0’,’ THE OUTPUT PLACE FOR IMMEDIATE TRANSITION ’,I5,
4 ’ IS ’,I5)
IF(NP.LE.0) GO TO 300
IF(NP.GT.NPLACE) GO TO 315
NOUIMM(I) = NOUIMM(I) + 2**(NP-1)
GO TO 315
300 CONTINUE
DO 51,I = 1,NTIMTR,
PRINT 60,I,NINTIM(I),NOUIMM(I)
60 FORMAT(’0’,’ FOR TIMED TRANSITION ’,I5,’ NINTIM= ’,I20,
4 ’ NOUIMM= ’,I20)
51 CONTINUE
DO 52,I = 1,NTIMTR
PRINT 61,I,NINTIM(I),NOUIMM(I)
52 FORMAT(’0’,’ FOR IMMEDIATE TRANSITION ’,I5,’ NINTIM= ’,I20,
4 ’ NOUIMM= ’,I20)
50 CONTINUE
PRINT 36
36 FORMAT(’1’, ’ THE AVERAGE FIRING RATES FOR THE TRANSITIONS ARE:’/
DO 37, I = 1, NTIMTR
READ 16,RTRANS(I)
37 FORMAT(F8.4)
PRINT 38,RTRANS(I)
38 FORMAT(’0’, ’ THE AVERAGE FIRING RATE OF TRANSITION ’,
2 I5,’ IS ’,F15.5)
35 CONTINUE
PRINT 65
65 FORMAT(’0’, ’ END OF DESCRIPTION OF THE GIVEN ’,
2 ’ GENERALIZED STOCHASTIC PETRINET’.)
PRINT 70
70 FORMAT(’0’, ’ DESCRIPTION OF THE INITIAL MARKING OF THE PETRINET.’)
DO 75, I = 1,NPLACE
READ 15,NTOKEN
PRINT 80,1,NTOKEN
75 FORMAT(’0’, ’ THE PLACE ’,I5,’ HAS ’,I4,’ TOKENS IN IT.’)
NTKTMP(I) = NTOKEN
74 CONTINUE
PRINT 81,(NTKTMP(J), J = 1,NPLACE)
81 FORMAT(’0’, ’ INITIAL MARKING IS : ’,205)
CALL FORMST(NPLACE,NTKTMP,ISTAT,ISTAT,ISTAT)
NSTATE(1,1) = ISTAT
NSTATE(1,2) = ISTAT
NSTATE(1,3) = ISTAT
PRINT 76,(NSTATE(1,J),J = 1,3)
76 FORMAT(’0’, ’ THE VALUE OF NSTATE(1) IS ’,3I20)
C NOW GENERATE THE VARIOUS REACHABLE STATES FOR BOTH TIMED AND IMMEDIATE
TRANSITIONS OF THE GIVEN GENERALIZED STOCHASTIC PETRI NET.

NUMST = 1
NIMM = 0
NTIME = 0

NEWFLG = 0
LIMIT = NUMST
IF(NUMST.EQ.1) ISTART = 1
PRINT 85,LIMIT

FORMAT('*',I5) LIMIT = ',L5
DO 90,M = ISTART,LIMIT
ENIMM = 0
ISTMONE = NSTATE(M,1)
ISTMTWO = NSTATE(M,2)
ISTMTHR = NSTATE(M,3)
CALL FNDMK(NPLACE,ISTMONE,ISTMTWO,ISTMTHR,NTKTMP)
DO 95,J = 1, NIMMTR
NTEMPJ1 = 0
NTEMPJ2 = 0
NTEMPJ3 = 0
INVEC = NIINMM(J)
IOUTVEC = NOUIMM(J)
IENIMM = ITRENA(NPLACE,INVEC,NTKTMP)
IF(IENIMM.EQ.0) GO TO 95
ENIMM = 1
CALL FNDNST(NPLACE,INVEC,IOUTVEC,NTKTMP,NTMPONE,NTMPTWO,NTMPTHR)
NTEMPJ1 = NTMPONE
NTEMPJ2 = NTMPTWO
NTEMPJ3 = NTMPTHR

CONTINUE
IF(ENIMM.EQ.0) GO TO 200
IF(NIMM.EQ.0) GO TO 210
DO 220,JK = 1, NIMM
IF((NITVAN(JK,1).EQ.NSTATE(M,1)).AND.
  2(NITVAN(JK,2).EQ.NSTATE(M,2)).AND.
  3(NITVAN(JK,3).EQ.NSTATE(M,3))) GO TO 225

CONTINUE

DO 230,LK = 1,3
NITVAN(NIMM,LK) = NSTATE(M,LK)
CONTINUE

DO 240,J = 1, NIMMTR
IF((NTEMPJ1.EQ.0).AND.(NTEMPJ2.EQ.0)
  2.AND.(NTEMPJ3.EQ.0)) GO TO 240
DO 250, I = 1, NUMST
IF((NTEMPJ1.EQ.NSTATE(I,1)).AND.
  2(NTEMPJ2.EQ.NSTATE(I,2)).AND.
  3(NTEMPJ3.EQ.NSTATE(I,3))) GO TO 240

CONTINUE

NUMST = NUMST + 1
DO 260,L = 1,3
NSTATE(NUMST,L) = NTEMPJL
CONTINUE

NEWFLG = 1
240 CONTINUE
   GO TO 90
200 ENATIM = 0
   DO 265, J = 1, NTIMTR
   NTEMP(J,1) = 0
   NTEMP(J,2) = 0
   NTEMP(J,3) = 0
   INVEC = NINTIM(J)
   IOUTVEC = NOUTIM(J)
   IENTIM = ITRENA(NPLACE,INVEC,NTKTMP)
   IF (IENTIM.EQ.0) GO TO 265
   ENATIM = 1
   CALL FNDNWST(NPLACE,INVEC,IOUTVEC,NTKTMP,NTMPONE,NTMPTWO,NTMPTHR)
   NTEMP(J,1) = NTMPONE
   NTEMP(J,2) = NTMPTWO
   NTEMP(J,3) = NTMPTHR
265 CONTINUE
   IF (ENATIM.NE.0) GO TO 100
   PRINT 105,M,(NSTATE(M,J),J=1,3)
   FORMAT('0', ' THE MARKING WITH STATE ',I4, ' IS A ' /,
5 ' TERMINAL MARKING'. AND NSTATE(M) = ',I40)
   PRINT 500,M,(NTKTMP(J), J = 1,NPLACE)
   GO TO 90
100 IF (NTIME .EQ. 0) GO TO 400
   DO 405-JK = 1,NTIME
   IF((NSTATN(JK,1).EQ.NSTATE(M,1)).AND.
2 (NSTATN(JK,2).EQ.NSTATE(M,2)).AND.
3 (NSTATN(JK,3).EQ.NSTATE(M,3))) GO TO 410
405 CONTINUE
   NTIME = NTIME + 1
   DO 415,LK = 1,3
   NSTATN(NTIME,LK) = NSTATE(M,LK)
415 CONTINUE
   DO 410,J = 1,NTIMTR
   IF((NTEMP(J,1).EQ.0).AND.(NTEMP(J,2).EQ.0)
2 .AND.(NTEMP(J,3).EQ.0)) GO TO 110
   DO 115, I = 1,NUMST
   IF((NTEMP(J,1).EQ.NSTATE(I,1)).AND.
2 (NTEMP(J,2).EQ.NSTATE(I,2))
3 .AND.(NTEMP(J,3).EQ.NSTATE(I,3))) GO TO 110
   CONTINUE
   NUMST = NUMST + 1
   DO 112,N = 1,3
   NSTATE(NUMST,N) = NTEMP(J,N)
112 CONTINUE
   NEWFLG = 1
110 CONTINUE
   IF (NUMST GT 8500) THEN
   PRINT 600,NUMST
   600 FORMAT ('0', ' THE VALUE OF NUMST IS ',I5, ' AND EXCEEDS THE ',
2 ' ARRAY RANGE OF NSTATE.')
   STOP
   ENDIF
C
C SEE IF ANY NEW STATES.
C
IF (NEWFLG.NE.0) THEN
  ISTART = LIMIT
  PRINT 700, ISTART
  700 FORMAT( '0', ' ISTART = ', I5)
  GO TO 85
ENDIF
ICOUNT = NTIME + NIMM
IF (NUMST .EQ. ICOUNT) GO TO 425
PRINT 430
430 FORMAT( '0', ' ERROR IN THE DETERMINATION OF REACHABILITY',
      '2 ', ' SET: ')/., ' THE TOTAL NUMBER OF STATES DO NOT TALLY WITH',
      '3 ', ' SUM OF NTIME AND NIMM STATES.' )
  PRINT 435, NUMST, NTIME, NIMM
  435 FORMAT( '0', ' NUMST = ', I5, ' NTIME = ', I5, ' NIMM = ', I5)
  STOP
425 PRINT 440, NUMST
440 FORMAT( '0', ' THERE ARE A TOTAL OF ', I5, ' STATES IN THE',
      '2 ', ' REACHABILITY SET OF THE GIVEN GENERALIZED STOCHASTIC',
      '3 ', ' PETRINET.' )
  PRINT 450, NTIME
  450 FORMAT( '0', ' THE GSP NET HAS ', I5, ' TANGIBLE STATES.' )
  PRINT 455, NIMM
  455 FORMAT( '0', ' THE GSP NET HAS ', I5, ' VANISHING STATES.' )
  PRINT 410, NUMST
120 FORMAT( '1', ' THERE ARE ', I5, ' STATES IN THE REACHABILITY',
      '4 ', ' SET OF THE GIVEN PETRINET.' )
  PRINT 460
  460 FORMAT( '1', ' THE MARKINGS AND VALUES FOR EACH STATE OF THE',
      '2 ', ' REACHABILITY SET OF THE GIVEN GSP NET IS AS SHOWN: ' )
  PRINT 465
  465 FORMAT( '0', ' THE STATE VALUE AND THE CORRESPONDING MARKING ',
      '2 ', ' FOR THE VARIOUS TANGIBLE STATES ARE AS GIVEN BELOW: ' )
  CALL LISTAT(NTIME, NPLACE, NSTANG)
  PRINT 470
  470 FORMAT( '0', ' THE STATE VALUE AND THE CORRESPONDING MARKING ',
      '2 ', ' FOR THE VARIOUS VANISHING STATES ARE AS GIVEN BELOW: ' )
  CALL LISTAT(NIMM, NPLACE, NSTVAN)
  ENDIME = SECOND()
  ELAPTIME = ENDTIME - STATIME
  PRINT 710, ELAPTIME
  710 FORMAT( '0', ' THE EXECUTION TIME IS: ', F15.6)
  STOP
END
SUBROUTINE LISTAT(NCOUNT, NPLACE, NSTARR)

C
C THIS SUBROUTINE LISTS THE VALUES OF VARIOUS STATE VALUES AND
C THEIR CORRESPONDING MARKINGS.
C
DIMENSION NSTARR(3000,3), NTOKEN(50)
DO 125, I = 1, NCOUNT
PRINT 130,1,(NSTARR(I,J),J=1,3)
130 FORMAT('0'," THE STATE NUMBER OF THE MARKING ',I5,
5' IS ',I320)
125 CONTINUE
C DETERMINE THE TOKENS IN VARIOUS PLACES FOR EACH STATE AND PRINT THEM...
C
DO 140,I=1,NCOUNT
ISTONE = NSTARR(I,1)
ISTWO = NSTARR(I,2)
ISTHR = NSTARR(I,3)
CALL FNDMKR(NPLACE,ISTONE,ISTWO,ISTHR,NTOKEN)
PRINT 160,1,(NTOKEN(J),J=1,NPLACE)
160 FORMAT('0'," STATE ',I3,' IS ',40I3)
140 CONTINUE
RETURN
END
SUBROUTINE FORMST(ICOUNT,NTKARR,ILOWNUM,IMIDNUM,IHIGNUM)
C THIS SUBROUTINE DETERMINES A STATE NUMBER FOR BASE 16 AND ON THE
C NUMBER OF TOKENS AT EACH PLACE AS GIVEN BY THE ARRAY NTKARR.
C
DIMENSION NTKARR(100),NHEXNUM(3)
DO 10,J = 1,3
NHEXNUM(J) = 0
10 CONTINUE
C DETERMINE THE STATE NUMBER FROM THE TOKEN INFO IN NTKARR.
C
K = 0
DO 15,I = 1,ICOUNT
NTOKEN = NTKARR(I)
IMINUS = I - 1
IMOD = MOD(IMINUS,12)
IF(IMOD.EQ.0) K = K + 1
NHEXNUM(K) = NHEXNUM(K) + NTOKEN*(16**IMOD)
15 CONTINUE
ILOWNUM = NHEXNUM(1)
IMIDNUM = NHEXNUM(2)
IHIGNUM = NHEXNUM(3)
RETURN
END
SUBROUTINE FNDMKR(ICOUNT,ISTONE,ISTWO,ISTHR,NTKARR)
C THIS SUBROUTINE DETERMINES THE VARIOUS TOKENS AT EACH OF THE PLACE
C IN THE GIVEN PETRINET AND STORES THIS INFORMATION IN THE NTKARR
C ARRAY.
C
DIMENSION NTKARR(100),NSTNUM(3)
NSTNUM(1) = ISTONE
NSTNUM(2) = ISTWO
NSTNUM(3) = ISTHR
DO 10,I = 1,ICOUNT
NTKARR(I) = 0
CONTINUE

FIND THE NUMBER OF TOKENS IN EACH PLACE AND STORE THE VALUE
IN THE NTKARR.

IMASK = 15
K = 1
ITMPNUM = NSTNUM(K)
DO 15, J = 1, ICOUNT
NTKARR(J) = ITMPNUM .AND. IMASK
NUMSHF = SHIFT(ITMPNUM, -4)
ITMPNUM = NUMSHF
IMOD = MOD(J, 12)
IF(IMOD.EQ.0) THEN
K = K + 1
ITMPNUM = NSTNUM(K)
ENDIF
15 CONTINUE
RETURN
END

FUNCTION ITRENA(ICOUNT, ITRIN, NTKARR)

THIS SUBROUTINE DETERMINES WHETHER A GIVEN TRANSITION IS ENABLED
FOR A GIVEN STATE AND ACCORDINGLY SETS A FLAG.

DIMENSION NTKARR(100)
IFLAG = 0
NUMB = 0
IONE = 1
IMARK = ITRIN
DO 20, K = 1, ICOUNT
IAND = IMARK .AND. IONE
IN(INAND.EQ.1) .AND. (NTKARR(K).NE.0)) NUMB = NUMB + 2**K
ISHFNUM = SHIFT(IMARK, -1)
IMARK = ISHFNUM
20 CONTINUE
IF(NUMB.EQ. ITRIN) IFLAG = 1
ITRENA = IFLAG
RETURN
END

SUBROUTINE FNOWNST(ICOUNT, INTR, NOUTR, NTKARR, ISTATE, ISTATE)

THIS SUBROUTINE DETERMINES THE NEW TOKEN DISTRIBUTION FOR THE ENABLED
TRANSITION AND FINDS THE MARKING STATE FROM THIS NEW TOKEN
DISTRIBUTION.

DIMENSION NTKARR(100), NTOKEN(100)
DO 10, I = 1, ICOUNT
NTOKEN(I) = NTKARR(I)
10 CONTINUE
IONE = 1
DO 15, K = 1, ICOUNT
INAND = INTR .AND. IONE
IF(INAND.EQ.1) NTOKEN(K) = NTOKEN(K) - 1
ISHPIN = SHIFT(NINTR, -1)
NINTR = ISHPIN
IOUTAND = NOUTR AND IONE
IF(IOUTAND.EQ.1) NTOKEN(K) = NTOKEN(K) + 1
ISHFOUT = SHIFT(NOULR, -1)
NOULR = ISHPFOUT
15 CONTINUE

C FIND THE NEW STATE FROM THE NEW TOKEN DISTRIBUTION GIVEN IN
C THE ARRAY NTOKEN.
C
CALL FORMST(ICOUNT, NTOKEN, IONETMP, ITWOTMP, ITHRTMP)
ISTONE = IONETMP
ISTWO = ITWOTMP
ISTHRE = ITHRTMP
RETURN
END
APPENDIX D:
Reachability set listing of SPN model SPN2A.
There are 72 states in the reachability set of SPN2A.

The various states in the reachability set are listed below:

<table>
<thead>
<tr>
<th>State</th>
<th>State Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1 0 0 1 0 1 1 0 0 1 1 0 1 0 0</td>
</tr>
<tr>
<td>2</td>
<td>0 0 0 1 1 1 1 0 1 1 0 0 0 1 0 0</td>
</tr>
<tr>
<td>3</td>
<td>0 1 0 1 0 1 1 0 0 1 1 0 1 0 0</td>
</tr>
<tr>
<td>4</td>
<td>1 0 0 1 0 1 1 0 1 0 1 0 1 0 0</td>
</tr>
<tr>
<td>5</td>
<td>1 0 0 1 0 1 1 0 0 0 1 1 1 0 0</td>
</tr>
<tr>
<td>6</td>
<td>1 0 0 0 0 0 0 0 0 0 1 1 0 1 1 0</td>
</tr>
<tr>
<td>7</td>
<td>0 0 0 1 1 1 1 0 1 0 1 0 1 0 0</td>
</tr>
<tr>
<td>8</td>
<td>0 0 0 1 1 1 1 1 0 0 0 1 1 1 0 0</td>
</tr>
<tr>
<td>9</td>
<td>0 1 0 1 0 1 1 0 1 0 1 0 1 0 0</td>
</tr>
<tr>
<td>10</td>
<td>0 1 0 1 0 1 1 0 0 0 1 1 1 0 0</td>
</tr>
<tr>
<td>11</td>
<td>1 0 0 1 0 1 0 1 0 0 0 1 0 0 0 0 1</td>
</tr>
<tr>
<td>12</td>
<td>0 0 0 0 0 1 0 0 0 0 0 1 0 0 1 0 0</td>
</tr>
<tr>
<td>13</td>
<td>0 1 0 0 0 0 0 0 0 0 1 1 0 1 1 0</td>
</tr>
<tr>
<td>14</td>
<td>1 0 0 0 0 0 0 0 1 0 1 0 1 1 0 0</td>
</tr>
<tr>
<td>15</td>
<td>1 0 0 0 0 1 1 0 1 0 1 1 1 0 0</td>
</tr>
<tr>
<td>16</td>
<td>0 0 0 1 1 1 1 0 0 1 1 1 0 0 0 1</td>
</tr>
<tr>
<td>17</td>
<td>0 0 0 1 1 1 1 0 0 0 1 0 0 0 0 1</td>
</tr>
<tr>
<td>18</td>
<td>0 1 0 1 0 1 1 0 0 0 1 0 0 0 0 1</td>
</tr>
<tr>
<td>19</td>
<td>1 0 0 1 0 1 0 1 0 0 0 1 0 0 0 0 1</td>
</tr>
<tr>
<td>20</td>
<td>1 0 0 1 0 1 1 0 0 0 1 0 0 0 0 1</td>
</tr>
<tr>
<td>21</td>
<td>1 0 1 0 1 0 0 0 0 0 1 0 1 0 0 0 1</td>
</tr>
<tr>
<td>22</td>
<td>0 0 0 0 1 0 0 0 0 1 0 0 1 0 1 0</td>
</tr>
<tr>
<td>23</td>
<td>0 0 0 1 1 1 1 1 0 0 0 0 0 1 0 0 1</td>
</tr>
<tr>
<td>24</td>
<td>0 0 0 1 1 1 1 1 0 1 0 0 0 0 1 0 1</td>
</tr>
<tr>
<td>25</td>
<td>0 0 0 1 1 1 1 1 1 1 0 1 0 0 0 1 0</td>
</tr>
<tr>
<td>26</td>
<td>0 0 0 1 1 1 1 1 1 1 1 1 0 1 0 0 1</td>
</tr>
<tr>
<td>27</td>
<td>0 0 0 1 1 1 1 1 1 1 1 1 1 0 1 0 0</td>
</tr>
<tr>
<td>28</td>
<td>0 0 0 1 1 1 1 1 1 1 1 1 1 0 1 0 0</td>
</tr>
<tr>
<td>29</td>
<td>0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 1 0</td>
</tr>
<tr>
<td>30</td>
<td>0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0</td>
</tr>
<tr>
<td>31</td>
<td>0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>32</td>
<td>0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>33</td>
<td>0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>34</td>
<td>0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>35</td>
<td>0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>36</td>
<td>0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>37</td>
<td>0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>38</td>
<td>0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>39</td>
<td>0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>40</td>
<td>0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>41</td>
<td>0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>42</td>
<td>0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>43</td>
<td>0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>44</td>
<td>0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>45</td>
<td>0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>46</td>
<td>0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>47</td>
<td>0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>48</td>
<td>0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>49</td>
<td>0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>STATE</td>
<td>50</td>
</tr>
<tr>
<td>STATE</td>
<td>51</td>
</tr>
<tr>
<td>STATE</td>
<td>52</td>
</tr>
<tr>
<td>STATE</td>
<td>53</td>
</tr>
<tr>
<td>STATE</td>
<td>54</td>
</tr>
<tr>
<td>STATE</td>
<td>55</td>
</tr>
<tr>
<td>STATE</td>
<td>56</td>
</tr>
<tr>
<td>STATE</td>
<td>57</td>
</tr>
<tr>
<td>STATE</td>
<td>58</td>
</tr>
<tr>
<td>STATE</td>
<td>59</td>
</tr>
<tr>
<td>STATE</td>
<td>60</td>
</tr>
<tr>
<td>STATE</td>
<td>61</td>
</tr>
<tr>
<td>STATE</td>
<td>62</td>
</tr>
<tr>
<td>STATE</td>
<td>63</td>
</tr>
<tr>
<td>STATE</td>
<td>64</td>
</tr>
<tr>
<td>STATE</td>
<td>65</td>
</tr>
<tr>
<td>STATE</td>
<td>66</td>
</tr>
<tr>
<td>STATE</td>
<td>67</td>
</tr>
<tr>
<td>STATE</td>
<td>68</td>
</tr>
<tr>
<td>STATE</td>
<td>69</td>
</tr>
<tr>
<td>STATE</td>
<td>70</td>
</tr>
<tr>
<td>STATE</td>
<td>71</td>
</tr>
<tr>
<td>STATE</td>
<td>72</td>
</tr>
</tbody>
</table>
APPENDIX E:

Reachability set listing of GSPN model GSPN2B.
THERE ARE A TOTAL OF 182 STATES IN THE REACHABILITY SET OF GSPN2B.

GSPN2B HAS 67 TANGIBLE STATES AND 115 VANISHING STATES.

THE FOLLOWING IS THE LISTING OF THE TANGIBLE STATES OF GSPN2B:

| STATE | 1   | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| STATE | 2   | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| STATE | 3   | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| STATE | 4   | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| STATE | 5   | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| STATE | 6   | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| STATE | 7   | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| STATE | 8   | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| STATE | 9   | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| STATE | 10  | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| STATE | 11  | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| STATE | 12  | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| STATE | 13  | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| STATE | 14  | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| STATE | 15  | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| STATE | 16  | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| STATE | 17  | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| STATE | 18  | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| STATE | 19  | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| STATE | 20  | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| STATE | 21  | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| STATE | 22  | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| STATE | 23  | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| STATE | 24  | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| STATE | 25  | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| STATE | 26  | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| STATE | 27  | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| STATE | 28  | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| STATE | 29  | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| STATE | 30  | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| STATE | 31  | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| STATE | 32  | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| STATE | 33  | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| STATE | 34  | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| STATE | 35  | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| STATE | 36  | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| STATE | 37  | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| STATE | 38  | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| STATE | 39  | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| STATE | 40  | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| STATE | 41  | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| STATE | 42  | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| STATE | 43  | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| STATE | 44  | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| STATE | 45  | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| STATE | 46  | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| STATE | 47  | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
STATE  48 : 0 0 0 1 0 1 0 1 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0
STATE  49 : 0 0 1 0 0 0 1 0 0 0 1 0 1 0 0 1 0 0 1 0 0 0 0
STATE  50 : 0 0 0 1 0 0 1 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 1
STATE  51 : 1 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0
STATE  52 : 1 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0
STATE  53 : 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
STATE  54 : 0 0 1 0 0 0 1 0 0 1 1 1 1 0 0 0 1 1 0 0 0 0 0
STATE  55 : 0 0 0 1 1 1 0 0 1 1 1 1 0 0 0 1 0 0 0 0 0 0 0
STATE  56 : 0 0 1 0 0 0 0 1 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0
STATE  57 : 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
STATE  58 : 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
STATE  59 : 1 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0
STATE  60 : 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
STATE  61 : 1 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0
STATE  62 : 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
STATE  63 : 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
STATE  64 : 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
STATE  65 : 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
STATE  66 : 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
STATE  67 : 1 0 0 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 0 0 0 0 0

THE FOLLOWING IS THE LISTING OF THE VANISHING STATES OF GSPN2B:

STATE  1 : 0 1 0 1 0 0 0 1 0 0 0 1 0 0 1 0 1 0 0 0 0
STATE  2 : 0 1 1 1 0 0 1 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 1
STATE  3 : 1 1 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
STATE  4 : 0 1 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
STATE  5 : 0 1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
STATE  6 : 0 1 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
STATE  7 : 0 1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
STATE  8 : 0 1 1 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
STATE  9 : 1 1 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
STATE 10 : 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
STATE 11 : 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
STATE 12 : 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
STATE 13 : 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
STATE 14 : 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
STATE 15 : 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
STATE 16 : 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
STATE 17 : 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
STATE 18 : 0 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
STATE 19 : 0 0 1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
STATE 20 : 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
STATE 21 : 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
STATE 22 : 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
STATE 23 : 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
STATE 24 : 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
STATE 25 : 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
STATE 26 : 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
STATE 27 : 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
STATE 28 : 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
STATE 29 : 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
| STATE 30 | 0 1 1 1 0 0 0 0 1 1 1 0 1 0 1 0 0 0 |
| STATE 31 | 0 1 0 1 0 0 1 0 1 1 0 0 1 0 1 0 0 0 |
| STATE 32 | 1 0 0 0 1 0 0 1 1 0 0 1 0 1 0 1 0 0 |
| STATE 33 | 1 0 0 0 0 1 1 0 0 1 1 0 0 0 1 1 1 0 |
| STATE 34 | 0 0 1 1 0 0 0 1 1 0 0 0 1 0 0 1 1 1 |
| STATE 35 | 0 0 0 1 0 1 1 0 0 1 0 0 1 1 0 0 1 1 |
| STATE 36 | 0 0 0 1 1 0 0 1 1 0 0 0 1 0 0 0 1 1 |
| STATE 37 | 0 0 0 0 1 0 0 1 1 0 0 0 0 1 1 0 0 1 |
| STATE 38 | 0 0 0 0 0 1 1 0 0 0 0 0 1 1 0 0 0 1 |
| STATE 39 | 0 0 1 0 1 0 0 1 1 0 0 0 0 0 1 1 0 0 |
| STATE 40 | 1 1 1 0 0 0 0 0 1 1 1 0 0 0 1 1 0 0 |
| STATE 41 | 1 1 0 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 |
| STATE 42 | 1 1 0 1 0 0 1 0 1 0 0 1 1 0 0 0 0 0 |
| STATE 43 | 1 1 0 1 0 0 1 1 0 0 1 1 0 1 1 0 0 0 |
| STATE 44 | 1 1 0 1 0 0 1 1 0 0 1 1 0 0 0 1 1 0 |
| STATE 45 | 0 1 1 1 0 0 0 0 1 1 0 0 1 1 0 0 0 1 |
| STATE 46 | 1 1 0 1 0 0 0 0 1 1 0 0 0 0 1 1 0 1 |
| STATE 47 | 1 1 0 0 0 0 0 0 1 1 0 0 0 1 1 0 1 0 |
| STATE 48 | 1 1 0 0 0 0 0 0 1 1 0 0 0 0 1 1 0 0 |
| STATE 49 | 1 0 0 0 0 0 1 1 0 0 0 1 1 0 0 0 1 1 |
| STATE 50 | 1 0 0 0 1 0 0 1 1 0 0 1 1 0 0 0 1 1 |
| STATE 51 | 0 0 1 1 0 0 0 0 1 1 0 0 1 1 0 0 0 0 |
| STATE 52 | 0 0 1 1 0 0 0 0 1 1 0 0 0 0 1 1 0 0 |
| STATE 53 | 0 0 1 1 0 0 0 0 1 1 0 0 0 0 0 0 0 1 |
| STATE 54 | 0 0 0 1 0 0 1 0 0 0 1 1 0 1 1 0 0 0 |
| STATE 55 | 0 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 |
| STATE 56 | 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 |
| STATE 57 | 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 |
| STATE 58 | 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 |
| STATE 59 | 0 0 0 0 0 1 1 0 0 0 1 1 0 0 0 0 0 0 |
| STATE 60 | 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 |
| STATE 61 | 1 1 0 0 0 1 0 0 0 1 1 0 0 1 1 0 0 0 |
| STATE 62 | 0 0 0 0 1 0 0 1 0 1 0 0 0 1 1 0 0 0 |
| STATE 63 | 0 1 1 1 0 0 0 0 1 1 0 0 1 0 0 0 0 1 |
| STATE 64 | 0 1 0 1 0 1 0 0 1 1 0 0 0 0 1 1 0 0 |
| STATE 65 | 0 1 0 1 0 0 0 0 1 1 0 0 0 0 0 0 1 1 |
| STATE 66 | 1 1 1 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 |
| STATE 67 | 1 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 1 1 |
| STATE 68 | 1 1 0 0 0 0 0 0 1 1 0 0 0 0 1 1 0 0 |
| STATE 69 | 1 0 0 0 0 0 0 0 1 1 0 0 0 0 1 1 0 0 |
| STATE 70 | 1 0 0 0 0 0 1 0 1 1 0 0 0 1 0 0 0 1 |
| STATE 71 | 0 1 1 1 0 0 0 0 0 1 0 0 0 0 1 1 0 1 |
| STATE 72 | 0 0 1 1 0 0 1 0 1 0 0 0 1 1 0 0 0 0 |
| STATE 73 | 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 |
| STATE 74 | 0 0 0 1 1 0 0 0 1 0 0 0 0 0 1 0 0 0 |
| STATE 75 | 0 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 0 0 |
| STATE 76 | 0 0 0 1 0 0 1 1 0 0 0 0 0 1 1 0 0 0 |
| STATE 77 | 0 0 0 1 0 0 1 1 0 0 0 0 0 1 1 0 0 0 |
| STATE 78 | 0 0 0 1 1 0 0 0 1 1 0 0 0 0 1 1 0 0 |
| STATE 79 | 0 0 0 1 1 0 0 0 1 1 0 0 0 0 1 1 0 0 |
| STATE 80 | 0 0 1 1 0 0 1 0 0 0 1 1 1 1 0 0 0 0 |
| STATE 81 | 1 1 1 1 0 0 0 0 0 0 1 1 1 0 0 0 0 0 |
| STATE 82 | 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1 1 0 0 |
| STATE 83 | 1 1 0 0 0 1 0 0 0 1 0 0 1 0 0 0 1 1 |