NOTICE

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

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

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

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

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

AVIS

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

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

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

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

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

Canada

COLORED GENERALIZED STOCHASTIC PETRI NETS FOR INTEGRATED SYSTEMS PROTOCOL MODELLING

by

MAN LI

A Thesis
Submitted to the
School of Graduate Studies and Research
in partial fulfillment of the requirements
for the degree of
MASTER OF APPLIED SCIENCE

Ottawa-Carleton Institute for Electrical Engineering
Department of Electrical Engineering
Faculty of Engineering
University of Ottawa

December, 1987

ISBN 0-315-46807-6

UNIVERSITÉ D'OTTAWA
UNIVERSITY OF OTTAWA

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to Dr. N. D. Georganas, not only for his excellent supervision, his financial support, but also for his kind help in many other aspects.

I would like to express my gratitude to the Chinese Government for their financial support during the first year of this research.

I would like to express my thanks to P. Gross for his help in running GSPNA, and to R. Kositpaiboon, L. Orozco and other colleagues for their useful discussions during this research.

Also, I would like to thank J. Burgess. Without his good management of the Unix Operating System, this research will not have been completed on time.

Finally, I would like to highlight the very special suppport from my family and many of my Chinese friends. They helped me in so many ways that it is not possible to list them here.

TABLE OF CONTENTS

CHAPTER I
INTRODUCTION

The recent advances in computer and communication technologies and the growing requirements for communications have made it inevitable that the different types of traffic such as data, voice, image, etc. will be integrated in the same network to provide a rich set of services.

The main characteristics of these integrated systems are clear: they must deal with different media , voice, data, etc. which have different requirements. They must also have layered functional protocols. The need for tools for the system performance evaluation becomes more and more apparent.

Performance modelling and evaluation refer to the following procedure: build a model which reflects the main factors determining system performance, obtain performance measures by the model and use these measures as estimates of performance of the real system. During the design and development phase of a system, modelling is the only way for predicting the system performance and for finding whether the designed system has a good performance at a reasonable cost. Even when a system is built up, modelling is still an appropriate tool that assists in understanding the behavior of the system and improving its performance. Modelling is, in general, less laborious, more flexible than experiments and more reliable than intuition.

As far as protocol performance is concerned, most previous work

concentrated on only one specific protocol layer. Layers above the one considered were usually modelled as workload sources, while those below were considered as transmission models. In this way, however, an important part of protocol was neglected, namely, resource contention from different layers (in multilayer protocols), cooperation, synchronization and concurrency between different layers, as well as the different behavior of digitized voice and a variety of data traffics.

With these problems in mind, we shall examine several possible ways for integrated system protocol performance analysis, describe their limitations, and then propose a better way. The proposed tool is CGSPN, an extention from CSPN. To make it more suitable for performance evaluation, we introduce two new concepts into it: "neutral tokens" and "priority firings". The extentions introduce mathematical complications which do not exist in either CSPN or GSPN. The underlying stochastic process associated with it becomes complex. Is it still a Markov Chain? We shall identify those problems and use the theory of stochastic processes to solve them. We shall also show that the extended CGSPN is still equivalent to a Markov Chain. In order to get numerical solutions for the proposed CGSPN, we build a PASCAL programme to extend the software package GSPNA which is used for analyzing GSPN. We also show how to implement the CGSPN to model integrated system protocols and how to obtain numerical results by the extended software package.

The organization of the thesis is as follows:

Chapter II reviews three ways in protocol evaluation, Queueing systems, Finite State Machines (FSM), and Petri Nets (PN) together with their strengths and weaknesses. Markov chains as an essential concept are

briefly reviewed at the beginning.

Chapter III first discusses PNs in details, particularly Stochastic Petri Nets(SPN) ,Generalized Stochastic Petri Nets (GSPN) and Colored Stochastic Petri Nets (CSPN). Colored Generalized Stochastic Petri Nets (CGSPN) are extended with the introduction of "neutral tokens" and "priority firings" and proposed as a tool for multilayer, integrated system protocol performance analysis. Problems associated with the propositions are identified and solved. A method for obtaining numerical results from CGSPN by enhancing GSPNA, a software package for analyzing GSPN, is given and a PASCAL program is built to extend the GSPNA. Methods for applying CGSPN to build the model of an integrated system protocol and for analyzing the model by the extended software package are also discussed.

Chapter IV describes an integrated data voice system (IDVS), particularly the protocols associated with it. This system will be analyzed by the extended CGSPN.

Chapter V builds the CGSPN model of the multilayer protocol of the IDVS described in Chapter IV and analyzes it by the extended software package. Results are discussed.

The thesis ends with Chapter VI, a conclusion about the extended CGSPN and suggestions for future study.

CHAPTER II
TOOLS FOR INTEGRATED SYSTEM PROTOCOL PERFORMANCE EVALUATION

In this chapter, three types of tools are reviewed as possible ways for integrated system protocol performance evaluation. They are: Queueing Systems, Finite State Machines and Petri Nets. Their strengths and weaknesses are discussed. Markov Chains as an essential concept are briefly reviewed at the beginning.

2.1 Markov Chains

Since Markov chains appear in this thesis several times, it is necessary to review some basic concepts of them.

A stochastic process is a family of random variables $\{X(t), t \in T\}$ defined on a given probability space, and indexed by the parameter t. The values assumed by the random variable X(t) are called states, and the set of all possible states forms the state space of the process.

If a stochastic process $\{X(t), t \in T\}$ satisfies the condition:

$$P\{X(t)<=x \mid X(t_n)=x_n, X(t_{n-1})=x_{n-1},\ldots,X(t_0)=x_0\}$$
$$= P\{X(t)<=x \mid X(t_n)=x_n\} \qquad t > t_n > t_{n-1} > \ldots > t_0 \qquad (2.1)$$

it is called a Markov process. $P\{X(t)<=x \mid X(t_n)=x_n\}$ is called transition

probability. The condition above is known as Markov property. Intuitively, a Markov process can be explained as a process where the future behavior only depends on the present state, not on the past procedure. In other words, the state $X(t_n)$ contains all necessary information about the past process history.

If the state space is discrete, the Markov process is called Markov Chain. The index t can be either discrete or continuous. As a result , there are discrete time Markov Chains and continuous time Markov Chains. If the transition probabilities are independent of time, then the Markov Chain is said to be homogeneous. In this thesis, only homogeneous Markov Chains are used.

For discrete time Markov Chains (DTMC), (2.1) becomes:

$$P\{X_{n+1}=x_{n+1} \mid X_n=x_n, X_{n-1}=x_{n-1},\ldots X_0=x_0\}$$
$$= P\{X_{n+1}=x_{n+1} \mid X_n=x_n\} \qquad \text{for all integer n}$$

$P\{X_{n+1}=x_{n+1} \mid X_n=x_n\}$ is the one step transition probability of the DTMC. In the homogeneous case, it is denoted as:

$$p_{ij} = P\{X_{n+1}=j \mid X_n=i\}$$

Define the transition probability matrix **P** as:

$$\mathbf{P} = [\, p_{ij} \,]$$

The steady state distribution of a DTMC is of particular interest.

If a DTMC is ergodic, then the steady state probability distribution exists. Let

$$\pi = \{ \pi_1, \pi_2, \ldots \pi_n \}$$

denote the steady state probability vector, where π_i is the probability of the DTMC being in state i. It is shown [TRIV 82] [ISAA 76] that π can be obtained by solving

$$\pi = \pi \mathbf{P}$$
$$\Sigma_i \pi_i = 1$$

For Continuous Time Markov Chains (CTMC). (2.1) becomes:

$$P\{X(t_{n+1})=x_{n+1} \mid X(t_n)=x_n, X(t_{n-1})=x_{n-1},\ldots,X(t_0)=x_0\}$$
$$= P\{X(t_{n+1})=x_{n+1} \mid X(t_n)=x_n\}$$

$$t_{n+1} > t_n > \ldots > t_0 \quad \text{for all integer n}$$

$P\{X(t_{n+1}) = x_{n+1} \mid X(t_n) = x_n\}$ is one step transition probability. In the homogeneous case, it is denoted as:

$$P_{ij}(\tau) = P\{ X(t+\tau) = j \mid X(t) = i\}$$

Define the transition rate matrix Q as

$$Q = [q_{ij}]$$

where

$$q_{ij} = \lim_{t \to 0} \frac{P_{ij}(t)}{t} \qquad i \neq j$$

$$q_{ii} = \lim_{t \to 0} \frac{P_{ii}(t)-1}{t}$$

Let π denote the steady state distribution of CTMC,

$$\pi = \{ \pi_1, \pi_2, \ldots \pi_n \}$$

then π can be obtained by solving: [TRIV 82]

$$\pi \mathbf{Q} = \mathbf{0}$$

$$\sum_i \pi_i = 1$$

It is proved [KLEI 75] that the time spent in CTMC states is an exponentially distributed random variable. If the time spent in a state has an arbitrary distribution but at the instants of state transitions the process behaves just like an ordinary Markov Chain, the points at these instants of state transition constitute an embedded Markov Chain. Hence, the statistics at these points may be obtained by solving the embedded Markov Chain.

2.2 Queueing Systems

Queueing networks are well known models of computer systems. Because they are capable of representing the main features of those systems.

A queue is a center, where customers come, wait in line for service, be served by servers according to certain service scheduling or queueing discipline, and depart after service. There are many service disciplines, for

instance, First Come First Serve (FCFS), in which customers are served according to the order of their arrival; Last Come First Serve (LCFS), in which the latest arrival is served first; no queue or Infinite Server(IS), in which infinite servers are assumed, hence customers receive service as soon as they arrive at the queue; Round Robin(RR), in which an interval of time is defined to be the quantum and customers are served in first come first served order as long as their service times do not exceed the quantum. When a customer's current service time(ignoring previous quanta) reaches the quantum, the service is preempted with state information saved for future resumption and the customer is placed at the end of the queue. When the quantum goes to zero, the RR discipline becomes processor sharing (PS), in which many customers may be served at the same time, every one served with equal rate, and the total rate being equal to the service rate.

The number of customers in the system constitutes a stochastic process. The properties of the process are determined by the interarrival time distribution, service time distribution and service scheduling. It is in general very difficult to obtain the transient behavior of the stochastic process. Instead, the steady state behavior of the process is often analyzed. By assuming exponential time distribution of the interarrival time and the service time, a continuous time Markov Chain is recognized. Each state of it represents the number of customers in the queue. The transition rate matrix **Q** can be obtained from the corresponding exponential time distribution. By solving:

$$\pi Q = 0$$
$$\sum_i \pi_i = 1$$

the steady state probability distribution of the number of customers in the queue is obtained. When either the interarrival time or the service time is not

exponential, i.e. general distribution, an embedded Markov Chain may be obtained. Hence, the probability distribution for the number of customers at specific points of time may be obtained by solving the embedded Markov Chain. Refer to [KLEI 75] for details.

Often, a single queue is not sufficient for modelling computer communication systems. Instead, a network of queues is needed. A network of queues consists of a set of service centers or queues. A customer enters the system, queues for service and, upon the departure from a given queue, he either proceeds to some other queues to receive additional service or leaves the system.

Customers may belong to different classes. The behavior of customers in each class is the same (e.g. same routing, same service time distribution, etc.). Customers of different classes may have different behaviors. A network is said to be open, if customers arrive and leave the network. A network is said to be closed, if customers neither leave nor arrive. A network may be closed with respect to some classes of customers and open with respect to other classes of customers. In this case, it is called a mixed network.

In a network of queues, the number of states of the corresponding Markov Chain may be high. If the number of queues is large, the Markov Chain may not be managable. Fortunately, imposing certain restrictions on the characteristics of the workload and on the operation of the system's resources, some networks possess straightforward solutions, which are called product form solutions.

Assume that the network contains N queues or service centers and R classes of customers. The equilibrium state probabilities have the general

form [BCMP 75]:

$$P(S) = C\, d(S)\, f_1(x_1)\, f_2(x_2) \ldots f_N(x_N)$$

where

S : the state of the system.

x_i : the configuration of customers at the ith service center.

d(S): a function of the state of the model.

f_i : a function that depends on the type of the ith service center,

C : normalizing constant.

Four types of service centers are considered:

1. First Come First Served(FCFS), in which the service time distributions must be identical and exponential for all classes of customers;
2. Processor Sharing(PS);
3. Last Come First Served(LCFS);
4. Infinite Server(IS).

For the last three service centers, the probability distributions of the service times should have rational Laplace transforms.

Each customer belongs to one class of customers while awaiting or receiving service at a service center, but may change classes and service centers according to a fixed probability at the completion of a service request. For open networks, state dependent arrival processes are allowed.

The above product form networks have been extended, and some of the restrictions have been relaxed[BOUD 85] [CONW 85]. Efficient solution

algorithms exist [BRUE 80] [REIS 80] [CONW 85] . However, the restrictions still hamper their applications.

Before obtaining further results as the average residence time in the queue, let us first review one important rule, Little's result [LITT 61].

Little's result says: the average number of customers in a queueing system is equal to the average arrival rate of customers admitted in that system, multiplied by the average time spent in that system.

$$N = \lambda T$$

where

N: the average number of customers

λ: the arriving rate

T: the average time spent in the system.

Note that this rule does not need any specific assumptions regarding the interarrival time and service time distributions. Further, this rule may be applied to any part of a system. It is worth mentioning that the word "system" in the rule may consist of several queues, not necessary only one queue, as long as a boundary is specified.

The average number of customers in a system may be obtained by:

$$N = \sum_i i \cdot P_i$$

where

N: average number of customers.

P_i: probability of having i customers in the system.

The arriving rate is given. Hence the average time spent in the system is obtained by Little's result as:

$$T = N / \lambda$$

where λ is the arriving rate.

A few papers address the problem of protocol performance modelling and evaluation by Queueing theory [MITC 86] [BRUN 86] [REIS 79]. Reiser has made a survey on the communication system models embedded in the OSI reference model [REIS 86]. Most of the studies consider only one specific layer, the layers above and below the one of concern being represented simply by workload source and transmission models for analytical tractability. For example, for the OSI Network layer and the Transport layer, queueing networks and multichain closed queueing networks have been used respectively.

Murata [MURA 87] presented a two layer performance model, which consists of a Media Access Control (MAC) layer submodel and a transport layer submodel. A polling model is applied to the token passing MAC layer, and the closed queueing network model is applied to the Transport layer. An iterative algorithm for the two layer model is proposed: at the beginning, the arrival rate λ' from Transport layer to MAC layer is set to some initial value. The mean message delay times are calculated in terms of the given λ' through the MAC model. The mean delay is used to calculate the throughput λ of the Transport layer in the Transport layer model. Then setting the arriving rate λ' to λ, and repeating the procedure again, another λ is obtained. The process is repeated over and over until the difference between $\lambda'$and λ is small enough. Mitchell [MITC 86] presented a three level Local Area model. His solution method is also iterative, similar to the one discussed

above. Iterative algorithms are a good way for handling the multilayer protocol problem. But with a separate model of each layer, it is not possible to consider the contention for one resource (e. g. the CPU) from different layers.

One weakness of queueing system models is the difficulty in modelling and analyzing synchronization or cooperation. Even if some special symbols have been introduced to represent the synchronization [SAUE 81], no algorithms exist for solving it. As a result, whenever synchronization appears in models, people go to simulation for help. Another weakness is that when the conditions for product form solutions are violated, one has to go back to the Markov Chain itself, which could be too large to handle.

2.3 Finite State Machines

Protocols for computer communication systems were first described in natural language. Later it was found that these descriptions often turned out to be ambiguous. As a result, several other tools were used. One of them is Finite State Machines(FSM).

In the FSM model, protocol processes are specified by a set of states S representing the states of the process, a set of messages M that can be sent from one process to another, a set of initial states O and a set of transitions T representing mappings of states.

FSM = (S, M, O, T)

where

$S = \{ s_1, s_2, \ldots s_m \}$

$M = \{M_1, M_2, ...M_n\}$

$O = \{O_1, O_2, ...O_m\}$

T is a mapping from one state to other states.

A few papers in the literature have presented ways for protocol performance evaluation through FSM [RUDI 83] [RUDI 85] [KRIT 85] [KRIT 86] [ENGE 86].

A technique for predicting protocol performance directly and automatically from the FSM specification was presented by Rudin [RUDI 83]. A global system state is built from the FSM of each process. This state consists of the states of the processes involved in the protocol. The elapsed time incurred by the traversal of each arc is specified beside that arc. When one of several paths is possible, the relative probabilities of these several paths are specified.

With that model, it is possible to obtain the time delay between any two states or the recurrent time to one state. The delays associated with each of the intervening arcs are simply summed up. If several paths are possible, paths would have to be weighted according to the probabilities of their occurrence:

$$\text{Average Delay} = \sum_{i=1}^{N} [P_i * D_{path_i}]$$

where

D_{path_i} : the delay of path i

P_i : the probability of path i occuring

N : the number of paths.

Later Rudin improved this algorithm [RUDI 85]. He recognized the

Markov like property that the future growth of a path is independent of the path's history. The improved algorithm is based on merging equivalent states, which needs to be developed during the course of the analysis. Refer to [RUDI 85] for details.

The method proposed by Rudin is easy to understand and easy to use, but it does not take into account resource contention.

Kritzinger proposed an analytical model for predicting the performance of implementation built according to the OSI basic reference model [KRIT 85] [KRIT 86]. The model uses the peer protocol standard of a layer as the reference description of an implementation of that layer.

His approach is the following:

Take the FSM for protocol specification, add the expected duration to each transition representing the time needed for that transition. When there exists more than one output from one state, probabilities should be assigned. Then identify active transitions or actions and passive transitons or delays. Active transitions are the ones which require time from the processor to execute in the implementation of the protocol. Passive transitions are the ones which simply represent a delay. The time taken by an active or a passive transition, respectively, is considered to be spent in the state from which that transition originates.

Two assumptions are made [KRIT 86]:

1. A path is selected at random, independent of any other paths selected and independently from one state to the next.
2. The time taken to execute a transition has an arbitrary distribution

which is independent of the time spent in any other transition.

A closed multiclass multichain queueing network is built with a processor sharing center modelling process contention at the processor and a delay center modelling times spent waiting for responses from the corresponding peer processes. Each individual transition of the protocol constitutes a different class and each layer of the architecture forms a closed chain. The network is solved by product form solution [KRIT 86] [LAVE 83]. Performance statistics such as queue lengths and response times at the processor as a function of processor speed are obtained.

As a matter of fact, Krizinger's model uses FSM as a bridge leading to queueing networks. The model does consider the resource contention and is a good technique for handling the performance measures for multilayer communication architectures. The weakness of the model is that apart from the contention for processor time by the processes from all layers, the model does not directly correlate individual events at neighbouring layers, or the cooperation between different layers, which however constitute an improtant part of protocol specification. Petri Nets which will be briefly reviewed in the next section circumvent this problem.

2.4. Petri Nets

Petri Nets are a graphical model developed by C.A. Petri [PETR 66]. A Petri Net consists of a set of places P, a set of transitions T, and a set of arcs A. Graphically, places are represented by circles and transitions by bars. Places may contain tokens, which are drawn as black dots. A marking is an assignment of tokens to the places. A PN state is a marking. For a formal definition of PN, see [PETR 66] or Chapter III.

A place is an input place of a transition if an arc exists from the place to the transition. A place is an output place of a transition if an arc exists from the transition to the place. When all the input places of a transition contain at least one token, it is enabled. An enabled transition can fire removing one token from all input places and placing one token in each output place. In this way, a new marking is reached.

PNs are widely accepted as a tool for the modelling of computer systems as well as many complex systems due to their capability of representing synchronization , concurrency and conflicts.

In ordinary PNs, there is no time concept involved. However, as long as performance is concerned, time plays an important role in system operation. Recognizing that PNs are powerful in representing synchronization and concurrency which are not easy to model with queueing systems, a lot of efforts have been made to add time into the PNs. One of them is the E-nets [NOE 73] which introduced a fixed time associated with each transition to specify the delay between the enabling and the firing of a transition. Another effort was made by Merlin [MERL 76]. He associated each transition with a maximum and a minimum firing times. Still another effort was made by Zuberek [ZUBE 80]. He associated firing frequencies and deterministic firing times with each transition. The net is further extended by Holliday [HOLL 85] to Generalized Timed Petri Nets (GTPN) by removing the restrictions of Zuberek's works. The result net is viewed as a discrete time, embedded Markov Chain. The time in a state is a deterministic function for each state of the net. A probability distribution is defined over the possible next states.

From the author's point of view, two kinds of extended PNs can be the

promising basis for protocol performance analysis. They are the Stochastic Petri Nets (SPN) and the Generalized Stochastic Petri Nets(GSPN). Their characteristic of merging the PNs, which are powerful in modelling synchronization and concurrency, with a stochastic process for obtaining the performance estimates seems to be attractive.

Stochastic PNs proposed by Molloy [MOLL 82] associate each transition in PN with an exponentially distributed random variable that expresses the delay from the enabling to the firing of the transition. He showed that the SPN is isomorphic to a continuous time Markov Chain because of the memoryless property of the exponential distribution. The power of SPN lies in the fact that by using the method of stages, it is possible to model transitions with generally distributed firing times provided these distributions have rational Laplace transforms. For a formal definition of SPN see [MOLL 82] or Chapter III.

Generalized Stochastic Petri Nets (GSPN) were proposed by Marsan [AJMO 84]. A GSPN contains two types of transitions: timed and immediate transitions. An exponentially distributed firing time is associated only with timed transitions, and immediate transitions fire in zero time. GSPNs are also equivalent to Markov Chains [AJMO 84]. Furthermore, the state space, or more precisely, the number of states in the corresponding Markov Chain, is reduced as a result of introducing immediate transitions. For a formal definition of GSPN, refer to [AJMO 84] or Chapter III.

PNs together with SPNs and GSPNs have found many applications in performance modelling. For instance, GSPN has been applied successfully in the analysis of multiprocessor systems [AJMO 86]. However, to the author's knowledge, although many papers use PN theory in protocol performancce analysis, e.g. [DIAZ 82] [WHEE 85] [MOLL 82], only a few of them got

analytical results. Among those who did have numerical results, their considerations were restricted to one layer of the protocol only, none of them tackling the multilayer protocol performance evaluation.

GSPN is well suited for performance analysis. However, it does not take into account the identity of tokens or allow classification of customers as in queueing systems. In the analysis of integrated systems, correctly identifying each medium, and hence correctly representing them by different classes are needed. Colored SPN (CSPN), which is able to identify different classes, is an extention to SPN [ZENI 85a].

In addition to all the elements in SPN, CSPN associates a set of colors with each place, with each transition as well as with tokens. The reachability tree for CSPN can be constructed following the firing rules. Since exponential firing time is assumed, the CSPN can also be viewed as a Markov Chain [ZENI 85a]. No author, however, has considered the priority firings for different classes.

In this thesis, we propose Colored Generalized Stochastic Petri Nets (CGSPN) as a tool for integrated system multilayer protoclol modelling. At the same time, the new concepts of "neutral tokens" and "priority firings" are introduced to make the tool more suitable for the analysis of multilayer protocol of integrated systems. It is shown that the resulted CGSPN is also a Markov Chain. A software package is extended to obtain numerical results from the extended CGSPN. Methods for using the CGSPN to build the model of an integrated system and obtaining the numerical results are discussed. An Integrated Data Voice System (IDVS), which is built in our lab, is analyzed by the extended CGSPN.

No matter of what kind of PNs, SPNs, GSPNs or CGSPNs we use, we

y used. From the author's point of view, the gain which this breakthrough quick growth of the state space of the underlying Markov Chain. This limits their application to systems of only a certain size. A breakthrough is needed before these tools can be widely used. From the author's point of view, the gain which this breakthrough brings to PNs will be similar as that product form solutions bring to the queueing system theory. In other words, a cost effective way should be found to solve those PNs. Since there are many basic differences between queueing networks and Petri Nets, the simple solution may not necessarily be a product form. It may be of other forms. The breakthrough in PNs lies in a deep understanding of the philisophy of PNs and a brilliant mind.

Recently, some authors work on the fast bounds for SPN and GSPN [MOLL 85] [BRUE 85], which is a bound estimation (e.g. throughput bound) for PNs without going into the details of analyzing them. It is worth mentioning that Lazar [LAZA 87] presented a kind of Petri Nets which have product form solutions. This work may be viewed as the frontier in the effort of finding a simple way to solve PN models.

CHAPTER III

Petri Nets

3.1 Petri Nets and Markov Chains

3.1.1 Standard Petri Nets

Petri Nets (PN) are an effective modelling tool for the description and the analysis of concurrency and synchronization in parallel systems exhibiting the cooperative actions of different entities [AJMO 86]. PNs were introduced by C. A. Petri in 1962 [PETR 66]. The theoretical problems associated with PN have been deeply investigated during the last twenty years. Today PN can be regarded as a formal structure for which a well-assessed theory has been developed and a wide range of application fields have been identified. The success of PN lies in the simplicity of the basic mechanism of the model, which is, however, paid for with the complexity of the description of large systems.

A Petri Net is composed of four parts [PETE 81]: a set of places P, a set of transitions T, a set of transition input arcs A_i and a set of transition output arcs A_o. In the graphical representation of PN, places are drawn as circles and transitions as bars. Arcs connect transitions to places and places to transitions. A place is an input to a transition if an arc exists from the place to the transition. A place is an output of a transition if an arc exists from the transition to the place. An example of PN is shown in Figure 3.1.

A formal definition of a Petri Net is thus the following [AJMO 86]:

$$PN = (P,T,A),$$

$$P = \{p_1,p_2,\ldots p_n\},$$

$$T = \{t_1,t_2,\ldots t_m\},$$

$$A_i \subset (P \times T)$$

$$A_o \subset (T \times P)$$

$$A = (A_i \cup A_o) \qquad (3.1)$$

A token is a primitive concept for PN. It is graphically drawn as a black dot in places. A <u>marking</u> m_i is an assignment of tokens to the places. A PN state is a marking denoted by $M = \{m_1,m_2,\ldots m_n\}$.

A formal definition of a <u>marked PN</u> is thus the following:

$$PN = (P,T,A,M_0),$$

$$P = \{p_1,p_2,\ldots p_n\},$$

$$T = \{t_1,t_2,\ldots,t_m\},$$

$$A_i \subset (P \times T)$$

$$A_o \subset (T \times P)$$

$$A = (A_i \cup A_o)$$

$$M_0 = \{m_{01}, m_{02},\ldots,m_{0n}\}. \qquad (3.2)$$

where m_{0i} denotes the number of tokens in place p_i in the initial marking M_0.

The rules concerning the excution of a PN are:

1. A transition is enabled when all of its input places contain at least one token.
2. An enabled transition can fire, thus removing one token from each input place and depositing one token in each output place.
3. Each firing of a transition modifies the distribution of tokens on places and thus produces a new marking for the PN.

In the following m_{pi} denotes the number of tokens in place p_i, $\#(p_a,t_b)$ is equal to 1 if an arc from place p_a to transition t_b exits (and to 0 otherwise); similarly $\#(t_a, p_b)$ is equal to 1 if an arc from transition t_a to place p_b exists (and to 0 otherwise). A transition t_b is enabled if

$$m_{pi} >= \#(p_i,t_b) \quad \forall p_i \, P, \tag{3.3}$$

and the result of the firing of t_b is a new marking $M' = \{m_1', \ldots m_n'\}$ defined as

$$m_{pi}' = m_{pi} - \#(p_i,t_b) + \#(t_b,p_i) \quad \forall pi \; P, \tag{3.4}$$

Tokens are atomic entities; i.e., the firing of a transition may disable other transitions by removing tokens from shared input places. Disabled transitions are said to be in conflict with the one that fires [AJMO 86].

A marking M' is defined to be immediately reachable from M if M' can be obtained by firing a transition enabled in M. The PN execution allows a sequence of markings $\{M_0, M_1, M_2, \ldots\}$ and a sequence of transitions $\{t_1, t_2,$

...} to be defined. A marking M" is defined to be reachable from M if there exists a sequence of transition firings that changes the PN state from M to M".

The reachability set $R(M_0)$ of a PN is the set of all markings that are reachable from M_0. A reachability tree represents the reachability set of a PN. The reachability tree can be constructed by starting from the initial marking M_0 and considering the markings immediately reachable from it. The same procedure must be followed considering the new markings. As the procedure is repeated over and over, every reachable marking will be eventually produced. Dead markings, i.e. markings in which no transition is enabled, as well as duplicate markings, i.e. markings previously appeared in the tree, constitute the frontier markings of the tree. The reachability tree of the PN in Figure 3.1 is shown in Figure 3.2.

3.1.2 Stochastic Petri Nets

Timing plays a fundamental role in the description and analysis of the behavior of any systems. However,standard PN does not include any time concepts,which means that with standard PN it is not possible to describe the time event. Hence, introducing time into PN is desirable.

The introduction of time into standard PN models allows the description of the dynamic behavior of systems, taking into account both the state evolution and the duration of each action performed by the system. Many of the most interesting parameters of system design, such as delay and throughput, and in general any time dependent behavior, can in this way be modeled [AJMO 86]. There are, however, different ways to introduce time into a standard PN.

One possibility is to associate with each transition a fixed time period, which indicates the delay between the enabling and firing of the transition. This kind of PN is called timed Petri Nets [ZUBE 80].

Another possibility is to associate random firing time with each transition. Using a stochastic model to represent PN which is capable to describe the synchronization and concurrency seems a very attractive way to obtain performance estimates of complex systems.

Stochastic Petri Nets (SPN), introcuced by M. Molloy [MOLL 81], are obtained by associating with each transition in a PN an exponentially distributed random variable that expresses the delay from the enabling to the firing of the transition.

A formal definition of an SPN is thus the following:

$$SPN = (P, T, A, M_0, L), \tag{3.5}$$

where P, T, A and M_0 are as in (3.2) and

$$L = \{l_1, l_2, ..., l_m\} \tag{3.6}$$

is a set of possibly marking dependent firing rates associated with the PN transitions. When necessary the dependence upon a given marking M of the firing rate of transition j will be denoted as $l_j(M)$.

Molloy [MOLL 81] showed that, SPNs are isomorphic to continuous time Markov Chains(MC). The extention of Petri Nets to stochastic Petri Nets allows the extraction of additional information on performance. The

countability of the markings and the memoryless property of exponential distributions are the key factors in allowing an isomorphism to exist.

The reachability set $R(M_0)$ of the PN constitutes the state space S associated with the SPN. The transition rate from state i, i.e. marking M_i, to state j is:

$$q_{ij} = \sum_{k \in H_{ij}} l_k \qquad (3.7)$$

where H_{ij} is the set of transitions enabled by marking M_i, whose firing generates marking M_j.

Molloy has also shown that, in some cases, with appropriate series and parallel expansions, by the method of stages, it is possible to model transitions with generally distributed firing times, provided that these distributions have rational Laplace transforms. In any case the extension from exponentially (memoryless) distributed firing times to generally distributed firing times must be done with great care.

An SPN is ergodic if it generates an ergodic CTMC. It is possible to show that an SPN is ergodic if M_0, the initial marking, is reachable from any M_i $R(M_0)$.

The analysis of the SPN may fall into two kinds:

If the equivalent Markov Chain has absorbing states, the expected number of steps until one of the absorbing states is reached may be calculated [ISAA 76], other quantitive estimates can be obtained from it.

Second, if the Markov Chain is ergodic, it is possible to compute the steady state probability distribution of markings by solving the matrix equation

$$\pi \mathbf{Q} = 0 \tag{3.8a}$$

with the additional constraint

$$\Sigma_i \pi_i = 1 \tag{3.8b}$$

where $\mathbf{Q}$ is the infinitesimal generator whose elements are obtained as explained in (3.7) and π is the vector of the steady state probabilities.

From the steady state distribution π it is possible to obtain quantitative estimates of the behavior of the SPN. The critical issue of it is to correctly identify the meaning associated with each π_i.

3.2 Generalized Stochastic Petri Nets

Occasionally, in a system, some events have large impact on system performance, whereas others have small impact. It is reasonable to associate random times only with the events that have the largest impact on system. The by-products that the number of states of the associated MC model is reduced and that compact models of complex systems can be constructed by clever thinking make this choice appealing.

Generalized Stochastic Petri Nets fulfill the above requirements. GSPN are obtained by allowing transitions to belong to two different classes :

immediate transitions and timed transitions [AJMO 86]. Immediate transitions fire in zero time once they are enabled. Timed transitions fire after a random, exponentially distributed, enabling time. Timed transitions are drawn as thick bars, and immediate transitions as thin bars. Firing rates are associated only with timed transitions, and they may depend on the GSPN marking. An example of GSPN is shown in Figure 3.3.

A formal definition of a GSPN is as in (3.5), where now the array L contains only m' elements, m' being the number of timed transitions in the GSPN.

Several transitions may be simultaneously enabled in a marking. If the set of enabled transitions H comprises only timed transitions, then transition t_i ($t_i \in H$) fires with probability [Appendix A]:

$$\frac{l_i}{\sum_{k \in H} l_k} \qquad (3.9)$$

If H comprises both immediate and timed transitions, then only immediate transitions can fire. When H comprises several immediate transitions, it is necessary to specify a probability mass function on the set of enabled immediate transitions 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 distribution is called a switching distribution. Different markings may engender a single random switch whenever they enabled the same set of immediate transitions upon which a single (possibly marking-dependent) switching distribution can be defined[AJMO 86].

3.2.1 Obtaining GSPN Steady State Probability Distribution

By examining the GSPN behavior as a function of time we can easily realize that it is equivalent to the time behavior of a stochastic process $\{X(t), t \geq 0\}$. A one to one correspondence exists between GSPN markings and stochastic process states. The process is observed to spend a nonnegative amount of time in markings which enable timed transitions only, while it transits in zero time through markings which enable immediate transitions. A state (or marking) of the former type is called tangible and a state (or marking) of the latter type is called vanishing.

For the purpose of performance evaluation in this thesis, the following assumptions will be made:

1. The reachability set is finite.
2. Firing rates are not time dependent.
3. The initial marking is reachable with a nonzero probability from any marking in the reachability set.

These assumptions further specify that the stochastic process is a finite state space, stationary (homogeneous), irreducible, and continuous-time stochastic process.

The analysis of the above stochastic process goes in two steps. First, ignore the concept of time, analyze the probability distribution of states. Second , incorporate time into each state to obtain the steady state probability distribution.

For the first step consider the set of states that the process enters

because of a transition out of a given state, an embedded Markov chain (EMC) can be recognized within the stochastic process.

Let S indicate the state space of the EMC (and of the stochastic process as well), and distinguish tangible and vanishing states within S so that it can be partitioned in the following way:

S = state space of the stochastic process; $|S| = K_s$;

T = set of tangible states in the stochastic process; $|T| = K_t$;

V = set of vanishing states in the stochastic process; $|V| = K_v$;

$S = T \cup V$

$T \cap V = 0$.

$K_s = K_t + K_v$.

The transition probability matrix U of the EMC can be written as follows [AJMO 86]:

$$\mathbf{U} = \mathbf{A} + \mathbf{B} = \begin{matrix} \\ K_v \\ K_t \end{matrix} \begin{matrix} \begin{matrix} K_v & K_t \end{matrix} \\ \begin{pmatrix} \mathbf{C} & \mathbf{D} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \end{matrix} + \begin{matrix} \begin{matrix} K_v & K_t \end{matrix} \\ \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{E} & \mathbf{F} \end{pmatrix} \end{matrix} \tag{3.10}$$

The elements of the matrix A can be obtained using the characteristics of random switches, and the elements of the matrix B can be obtained using the firing rates of timed transitions.

The solution of the system of linear equations

$$\mathbf{p} = \mathbf{pU} \tag{3.11}$$

$$\Sigma_i p_i = 1$$

in which **p** is a row vector representing the stationary probability distribution of the EMC, can be interpreted in terms of numbers of transitions performed by the EMC. Indeed, 1 / p is mean recurrence time for state i measured by the number of transitions.[ISAA 76].

By selecting one of the states of the EMC as a reference state for the chain, and assuming, without loss of generality, that it is a tangible state (say state i), the mean number of visits to state j between two subsequent visits to state i is obtained as [ISAA 76]:

$$V_{ji} = p_j / p_i . \qquad (3.12)$$

The computation of the steady state probability distribution of the stochastic process can be obtained by reintroducing the concept of time by means of the average sojourn time in each state (E [W_i], $i \in S$). Using the definition of immediate transitions, or equivalently that of vanishing states, we can write

$$E[W_i] = 0, \quad i \in V. \qquad (3.13)$$

The computation of the average sojourn times in tangible stats requires the definition of the set

Hi = { (timed) transition enabled by tangible state i}, $i \in T$,

so that

$$E[W_i] = \frac{1}{\sum_{j \in \pi_i} l_j} \qquad i \in T \tag{3.14}$$

The mean amount of time spent by the stochastic process in returning to the reference state i (mean cycle time) is thus given by [AJMO 86]:

$$C_i = \sum_{j \in S} V_{ji} E[W_j] = \sum_{j \in T} V_{ji} E[W_j] \tag{3.15}$$

where $V_{ji}E[W_j]$ is the mean time spent by the stochastic process in state j during a cycle. The average fraction of time spent by the stochastic process in each of its states can be computed as:

$$\pi_j = \frac{V_{ji} E[W_j]}{C_i} \qquad j \in S \tag{3.16}$$

The steady state probability distribution of the stochastic process can finally be written as

$$\pi_j = \begin{cases} 0, & j \in V; \\ \frac{V_{ji} E[W_j]}{C_i}, & j \in T \end{cases} \tag{3.17}$$

Given the conditions specified at the beginning of the section, it can be shown that this solution method is equivalent to a simple extension of the method originally proposed by Molloy to handle similar cases [AJMO 86].If all immediate transitions are replaced by timed transitions characterized by very high firing rates proportional to an arbitrary value x, the GSPN reduces to a standard SPN and Molloy's solution method applies. If an explicit solution expression for the probability distribution of this standard SPN (expressed in terms of x) is obtained, the stationary probability distribution

of the original GSPN can be obtained by taking the limit with x going to infinity. Since most practical cases involve GSPN with a large state space, an explicit expression of the solution in terms of x is usually not easy to obtain, and the practical approach suggested by Molloy of numerically solving the problem by assuming x to be very large and arbitrarily setting to zero those probabilities that appear exceptionally small has numerical problems.

The solution method proposed above is computationally acceptable. However, this method requires the computation of the mean number of visits to each vanishing state, which dose not increase the information content of the solution. Moreover, vanishing states not only require useless computations, but, by enlarging the size of the transition probability matrix U, make the computation of the visits expensive and in some cases even impossible to obtain.

Ajmone [AJMO 86] proposed a compact way which only computes the total transition probabilities among tangible states. This compact solution is adopted by GSPNA, a software package which we will discuss in the next section.

3.3 GSPNA: a Software Packet for Solving GSPN

GSPNA is a software package developed by Marsan [AJMO 85] for the steady state analysis of GSPNs. GSPNA has an interactive user interface that helps the analyst define the GSPN model of the system. The output of this first module is passed to the core of the analyzer that examines whether the Markov model is irreducible , and that computes the steady state solution of the original model.

GSPNA is organized into three modules that correspond to different

phases of the analysis of a real system: DESCRIPTION, SOLUTION, and USER FUNCTION. Figure 3.4 shows a block diagram of the software package. The following is a detailed description of the three modules.

DESCRIPTION is a interactive module, that helps the analyst define and modify the structure of a GSPN model. Executing DESCRIPTION, the user is guided in the specification of his model. If the model is new, the user must specify places, transitions, arcs, names of places and transitions. The specification of the associated input and output places for each transitions are also required. When the transition is declared to be of type TIMED, the associated firing rate needs to be introduced. When a transition is declared to be of type IMMEDIATE, the associated probability of firing needs to be given. Both these quantities may be marking dependent. In this case, the definition of appropriate external functions is required. A random switch among immediate transition is completely characterized by a set of immediate transitions simultaneously enabled by a marking. The transitions belong to the same group. The specification of all such groups is required at the end of the input description. This information is added to the input to simplify the construction of the stochastic model isomorphic to the GSPN. The description is completed when the initial marking is provided [AJMO 86].

SOLUTION is the core module that verifies the correctness of the graph model by building the reachability tree, maps it into the corresponding stochastic model, and finally computes the model solution in terms of the steady state distribution of tokens in each place of the GSPN.

Once the description of the model is completed, the file is passed to the SOLUTION module that performs the analysis of the GSPN. SOLUTION first reads the file. With the initial marking, it examines one-by one the timed transitions to see if they are enabled by this marking. With the enabled ones, it

goes to find the immediately reachable markings. If the immediately reachable marking is vanishing, it continues to find the next immediately reachable marking until a tangible marking is found. In this way , it constructs the first step in reachable tree. Then, it takes another new marking, goes through all transitions as before, but when it finds a marking, it first does a search to see whether this marking already exists or not. If not, a new "leave" is added to the tree. This procedure goes over and over until the tree is completely built. During this first phase of the analysis, the presence of absorbing states are identified, tangible and vanishing states are seperated. The SOLUTION adopts a compact method to obtain the transition probability matrix of the reduced embedded Markov chain directly, without sorting the larger transition probability matrix of the embedded Markov chain. When loops among vanishing states are recognized, the corresponding part of the original transition probability matrix is stored so that, later on, a matrix inversion is performed to obtain the desired reduced transition probability matrix.

The solution of the system of linear equations that yields the steady state probability distribution of markings is computed with the Gauss-Seidel iterative technique.

During the different solution phases, the program asks the user information of the level of detail of the desired output. The data produced during the execution of this module are stored in an output file, which can then either be printed, or used as a basis for the production of more specific reports.

USER FUNCTION is a template for the external definition of marking dependent firing rates and switching probability distributions. To simplify the specification of these functions, a set of built in functions that operate on

marking components is provided. Every time external functions of this type need to be defined for a GSPN model, the USER FUNCTION module must be separately compiled prior to the solution of the model. All the marking dependent functions of a GSPN must be defined inside the same USER FUNCTION module. They may be listed within a PASCAL CASE statement in which the entry numbers represent the corresponding transition numbers.

Inside the module it may be convenient to include a library of functions developed by the user to simplify the definition of marking dependent rates and probabilities for specific applications [AJMO 85].

3.4 Colored Generalized Stochastic Petri Nets.

GSPN is a useful modelling tool for the analysis and evaluation of the performance of computer systems. It is well suited for the representation of the behavior of computer systems including such features as synchronization, concurrency and conflict phenomena. However, it does not take into account the identity of tokens or the classification of customers as defined in Queueing Networks [KLEI 75]. On the other hand, classes are often essential in system performance modelling, especially for integrated systems. Colored Generalized Stochastic Petri Nets (CGSPN) can be viewed as an effort to compensate for the above limitations. Besides, it leads to a simple representation of systems.

Let us first examine Colored Stochastic Petri Nets(CSPN) proposed by ZENIE [ZENI85a]. A CSPN associates a set of colors with tokens indicating the identities of tokens. Places and transitions have also attached a set of colors. The color attached to a token may be changed by a transition firing.

The definition of a CSPN is as follows:

$CSPN = (P, T, C, A_i, A_o, M_0, L)$

where:

P is a set of places;

T is a set of transitions;

C is a color-function defined from PUT into a non-empty finite set.

A_i (or A_o) is a forward (or backward) incidence application defined

on $P \times T$ (or $T \times P$) such that $A_i(p,t)$ (or $A_o(t,p)$) is a member of $[C(t) \multimap C(p)]$ for all pairs $(p,t) \in P \times T$, where $[A \multimap B]$ denotes the set of functions from A to B;

M_0, called initial marking, is a function defined on P such that $M_0(p) \in C(p)$ for all $p \in P$

$L = \{l_{ij}\}$ $i \in T$, $j \in C$, is a set of color dependent firing rates associated with each transitions.

(3.18)

Exponential firing times are assumed. The execution of a CSPN is as follows:

1. A transition of color i is enabled when all of its input places specified by A_i contain at least one token of the corresponding color.
2. An enabled transition can fire, thus removing one token of the color specified by A_i from each input place and placing one token of the color specified by A_o in each output place.
3. Each firing of a transition modifies the distribution of tokens on places and thus produces a new marking for the CSPN.

We take one example from Zenie [ZENI 85a] to illustrate the concepts of CSPN, and we will further use it in the next section to illustrate the "unfolding" procedure.

The example is the"philosophers dining" problem. As shown in Figure 3.5, five philosophers, who alternately think and eat, sit around a table. To eat, a philosopher needs two forks, and he is allowed to use the two

forks nearest to him. However, there are only five forks on the table. Hence, two neighbours cannot eat at the same time.

The CSPN model is shown in Figure 3.6. Places "think" and "eat" as well as transitions "take forks" and "put down forks" are associated with colors 1 to 5 representing the states and the actions of the five philosophers. The place "free forks" is associated with a set of colors representing the five forks. To distinguish them from the identities of philosophers, let us give them the colors from 6 to 10. The firing rates of a transition could be different for each color. Initially, there are five tokens of colors 1 to 5 in place "think" and five tokens of colors 6 to 10 in place "free forks". The firing of transition "take forks" in color i (i = 1, 5) will absorb a token of color i from place "think" and two tokens of color (i+5) and color $(i+6)_{mod10}$ from place "free forks" and put one token of color i into place "eat" indicating that philosopher i is in the state of eating. The firing of transition "put down forks" in color i will absorb a token of color i from place "eat" and place a token of color i into place "think" and two tokens of color (i+5) and color $(i+6)_{mod10}$ into place "free forks" indicating that philosopher i has finished eating and is in the state of thinking. This example also shows that by introducing colors into PN, system description becomes more concise.

Zenie showed that there also exists an isomorphism between CSPN and a Markov Chain due to the memoryless property of the distribution of firing delays. He further showed that if the CSPN is bounded and live, and if the initial marking is reachable from any other markings, and if the firing rates are finite, then the Markov Chain is ergodic. [ZENI 85a] [ZENI 85b].

For the convenience of performance modelling, CSPN is extended to Colored GSPN, separating the transitions in CSPN into immediate transitions

and timed transitions. Immediate transitions fire in zero time once they are enabled. Timed transitions fire after a random, exponentially distributed, enabling time. Firing rates are associated only with timed transitions, and they may depend on the CGSPN marking.

A formal definition of CGSPN is as in (3.18), where now the array L contains only the number of timed transitions in CGSPN.

In the same way as in GSPN, markings in CGSPN are divided into vanishing markings and tangible markings. Vanishing markings are the ones by which immediate transitions are enabled, whereas tangible markings are the ones by which only timed transitions are enabled.

The extention from CSPN to CGSPN makes it possible to construct compact models for complex systems and reduce the underlying Markov states. However, at the same time it introduces some problems which are not encounted in any other kind of PNs.

One of them is that if a transition is timed in one color and immediate in another color, how to represent it graphically ? we suggest:

a. If a transition is a timed transition for all the colors, it is drawn as a thick bar.
b. If a transition is an immediate transition for all colors, it is drawn as a thin bar.
c. If a transition is a timed one in terms of some colors whereas an immediate one in terms of other colors, it is drawn as a dotted thick bar associated with a specification of the color dependent transition types.

An example of CGSPN is shown in Figure 3.7.

Another problem is that one transition may be simultaneously enabled in several colors by a marking M. What is the probability that a specific transition of a certain color fires?

Proposion 3.1:

a. If the set of enabled transitions H consists of only timed transitions, then transition t_i ($t_i \in H$) of color j fires with probability [Appendix A]:

$$\frac{l_{ij}}{\sum_{k \in H} \sum_{m \in Q_k} l_{km}} \qquad (3.19)$$

where Q_k comprises the set of colors for which transition k has been enabled.

b. If H comprises both immediate and timed transitions, only immediate transitions can fire.

c. If H comprises more than one immediate transitions, a probability mass function should be specified concerning the selection of firings.

We emphasize that a transition may be immediate in one color and timed in another color. If both colors are enabled, only the immediate one can fire.

The CGSPN is particularly well suited for quantitative analysis of models, especially for those systems with different classes behaving in a

similar way. The models constructed using this formalism are more concise, easier to understand and easier to use.

For the purpose of performance analysis in this thesis, two extentions to CGSPN are introduced.

The first concept is "neutral tokens". Occasionally, we would like to explicitly represent resources which may be accessed by every class. Neutral tokens are used for this purpose. After the introduction of neutral tokens, a transition is enabled if each of its input place contains at least one token of the required color or one neutral token.

An example of CGSPN with neutral tokens is given in Figure 3.8. For the convenience of reference, we define:

a. A place into which only neutral tokens may go is defined as a neutral place, other places are referred to as color places.
b. A transition which is only enabled by neutral tokens is defined as a neutral transition, other transitions in CGSPN are referred to as color transitions.

In Figure 3.8, places p1 and p2 are neutral places, and transitions t1 and t6 are neutral transitions. Transition t2 may fire in color i if there is a token of color i in place p3 and a neutral token in place p2. The firing of transition t2 will absorb the neutral token and the color i token and put a token of color i into place p5. Other transitions may fire when all of their input places contain at least one token of the same class. Note that we allow different color tokens to have different routings.The firing of transition t4 in color i will put back a neutral token to place p1, and if $i \leq m$ (refer to the figure), a token of color i will be put into place p4, otherwise, it will be put

into place p6.

The second concept we introduce is " priority firings" for different color tokens. The distinguished point for CGSPN is the ability to identify different classes. As a result, besides different routings for different color tokens, it is reasonable to allow different color transitions to have different priorities concerning transition firings.

Recall that in ordinary CGSPN if a set of timed transitions are enabled in a marking in several colors, say transition t_i is enabled in more than one colors, the probability that t_i of color j fires is as in (3.19). Now if different priority levels are assigned to each color, although there may be many transitions enabled by marking M, only the one of the colors which has the highest priority may fire.

Proposition 3.2:

When the set of enabled transitions are timed only, if color j is the highest priority color among those enabled transitions, and t_i of color j is enabled, then

a. t_i of color j will fire with probability [appendix A]:

$$\frac{l_{ij}}{\sum_{k \in H'} l_{kj}} \qquad (3.20)$$

where H' comprises all the enabled timed transitions of color j.

b. As a result of introducing priority firings into the net, transitions of color j fires with a higher probability because of its higher priority.

In fact, since

$$l_{ij} > 0$$

hence:

$$\sum_{k \in H_i} l_{kj} < \sum_{k \in H} \sum_{m \in Q_k} l_{km}$$

Therefore

$$\frac{l_{ij}}{\sum_{k \in H_i} l_{kj}} > \frac{l_{ij}}{\sum_{k \in H} \sum_{m \in Q_k} l_{km}}$$

In ordinary CGSPN, if several immediate transitions are enabled by marking M, a probability density function should be specified in order to select the one which fires. This selection is among all colors. After introducing " priority firings", the probability density function is specified within each color, since among the enabled transitions, only the ones with the highest priority color may fire.

One point which requires clarification is that priority is considered in timed transitions and in immediate transitions separately. When both immediate and timed transitions are enabled, only the immediate one may fire. Priority and probability mass function are then applied to those immediate transitions in deciding which one may fire.

Proposition 3.3:

The underlying stochastic process of the CGSPN with extentions to "neutral tokens" and "priority firings" is still a Markov Chain.

In fact, neutral tokens behave the same way as color tokens except for that they match any color. Hence the Markov property is not affected. "Priority firings" means that a transition of color i enabled in a marking can fire only when all the currently enabled transitions (including itself) are not enabled in colors with higher priority than color i. Consider an CGSPN with a marking M in which several timed transitions are simultaneously enabled with different color priorities. One of the transitions enabled with the highest priority color among them will fire. Then the CGSPN reaches a new marking M'. Those transitions which were already enabled in the previous marking M, but did not fire may still be enabled. Due to the memoryless property of the exponential distribution, the residual life distribution is equal to the firing distribution, i.e. exponential. Hence the activity associated with those transitions appears to be the same as newly enabled ones. This shows that the future evolution only depends on the present state not the past procedure. Therefore, the net result from the introduction of "priority firings" and "neutral tokens" is still Markovian.

In practical cases, things may be a little bit different. For example, priority firings may be only associated with some transitions, not necessary all the transitions. No matter what changes concerning priority may happen, the key is, whenever several transitions are enabled by one marking, to first identify which transitions can fire in which color according to the priority rules. If immediate ones can fire, we refer to a random switch specified concerning the situation. If only timed ones can fire, we apply (3.19), where H should be the set of all the transitions which may fire, and Q_k should be the set of all the colors for which transition k may fire at this time.

The main features of the CGSPN used in this thesis for protocol performance evaluation are summarized as follows:

-- a set of places;
-- a set of transitions, which can be either timed or immediate;
-- a set of colors associated with each place and each transition;
-- a set of colored tokens;
-- a set of neutral tokens;
-- a set of color dependent firing rates associated with each timed transition and a set of color dependent probabilities of firing associated with each immediate transition ;
-- color dependent arcs connecting places to transitions and transitions to places, which indicate the routing of tokens;
-- firing priority rules concerning the firing of transitions in each color.
-- different color tokens may have different routing, a transition may be an immediate one in one color and a timed one in another color.

Up to now, we have defined the concepts associated with the extended CGSPN. At the same time, we identified and solved the problems which are unique to the extended CGSPN. The next question is how to use the extended CGSPN as a tool to build the protocol performance model of integrated systems. We suggest the following steps:

1. Totally understand the protocol specification.
   Find precisely how many layers it has, what the functions of each layer are, how the different layers cooperate, how the sending side and the receiving side cooperate, and if there are priority rules concerning each media.
2. Find out the significant events in the execution of the protocol. Hence, identify the states of process.
   For example, "waiting for acknowledgement" could be one state.
3. Find out the primitives which cause the state transitions.
   For example, receiving acknowledgement may cause the sending

side to change from "waiting" state to "preparing another message" state.

4. Find out the statistical characteristics of different media. Decide how to represent them.

   Find exactly how media are represented. For example, a data message could be sent as a whole message or it could be broken down into small packets. Next find out their statistical estimates and decide whether represent them by exponential distributions or by stages of exponential distributions.

5. Find out the mean times needed for the execution of the primitives. These will be the mean exponential firing times of transitions.

6. Build the model:

   a. represent states as places.

   b. represent primitives as transitions.

   c. assign colors to places and transitions to distinguish different media.

   d. represent common resources as neutral tokens.

   e. represent different media as different colored tokens.

   f. make a list about the priority levels among all the transitions of all colors.

   g. assign input and output places for each transition according to the relationship between states and primitives;

   h. represent the transmitting and receiving phase by places and transitions according to step 4.

Now, the model is built. The next section will discuss how to solve the model by an extended software package. The method for extending GSPNA is discussed first.

In chapters IV and V, CGSPN will be applied to an Integrated Data

Voice System protocol performance evaluation. Chapter IV will highlight steps 1 to 3, while Chapter V will highlght steps 4 to 6.

3.5 CGSPNA: an enhancement to GSPNA

A Colored Generalized Stochastic Petri Net is a concise way for constructing a performance model. Nevertheless, it is not completed until a way to solve the model is found.

There are at least two ways for solving a CGSPN model. First is the "colored" way consisting of finding the reachability tree of the colored net and solving the associated Markov Chain.

The second is the "decolored" way. That is unfolding the color net into an ordinary Generalized Stochastic Petri Net and then solving the GSPN. Since the software package GSPNA is available, we have chosen the "decolored" way. DESCRIPTION is modified with a built in program which transforms a CGSPN into a GSPN. SOLUTION and USER FUNCTION are modified to implement priority firings.

The block diagram of the PASCAL program for transforming a CGSPN into a GSPN is shown in Appendix B. After we discuss the method for unfolding, we will give examples for demonstration. The principles concerning the unfolding are as follows:

1. A color place is replaced by a set of places, one for each color associated with it. A neutral place remains as one place. This place transform is realized by the procedure placetransform.
2. A color transition is replaced by a set of transitions, one for each color associated with it. A neutral transition remains as one

transition. This transition transform is realized by the procedure transtransform.

3. The arcs which connect places to transitions or transitions to places are transformed as follows:
   a. The arc which connects a neutral place to a color transition is replaced by a set of arcs which connect the neutral place to the transitions which replace the original color one. In the same way, the arc which connects a color transition to a neutral place is replaced by a set of arcs which connect the transitions which replace the original color one to the nertral place.
   b. The arc which connects a neutral transition to a color place is replaced by a set of arcs which connect the neutral transition to the places which replace the original color one. In the same way, the arc which connects a color place to a neutral transition is replaced by a set of arcs which connect the places which replace the original color one to the neutral transition.
   c. The arc which connects a neutral place to a neutral transition or a neutral transition to a neutral place remains as one arc with the same direction.
   d. The arc which connects a color place to a color transition is replaced by a set of arcs. The assignment of the relationships between the places which replace the original color place and the transitions which replace the original color transition should be according to the color correspondence functions associated with each transition.

   This input and output arc transformation is realized by the procedure getinoutput.
4. Firing rate for a timed transition or the probability of firing for an immediate transition of a certain color goes to the specific transition which represents it. This is realized by the procedure getattribute

embedded in the procedure getinoutput.

5. The immediate transitions which are in the same group are still in the same group after the transformation. This is realized by the procedure getgroup.
6. Initial tokens are distributed with each color to the corresponding places. This is realized by the procedure getinitial.
7. All the attributes associated with the result GSPN (e.g. initial tokens in each places, firing rates, etc.) as well as the one to one correspondence between the colored net and the decolored net are recorded into a file. this is realized by the procedure writefile.

The file created by DESCRIPTION is passed to the SOLUTION.

To illustrate the unfolding method discussed above, let us take two examples. The first is the "philosophers dining" problem. Although it is a SPN without neutral tokens, it is no less typical in highlighting some of the above steps. The unfolding result is shown in Figure 3.9.

The place "think" is replaced by five places thi (i = 1,5), each representing one philosopher who is in thinking state. Similarly, place "eat" is replaced by five places ei (i = 1,5) . Place "free forks" is replaced by five places ffi (i = 1,5), each representing one free fork.

The transitions "take forks" and "put down forks" are replaced respectively by five transitions ai (i = 1,5) and bi (i = 1,5).

The relationships between the places and transitions are assigned according to the color correspondence.

Firing rates δ_i (i = 1,5) and μ_i (i = 1,5) go to the specific transitions

which represent them.

Finally, initial tokens are distributed with each color to the corresponding places.

The second example is the unfolding of the CGSPN with neutral tokens in Figure 3.8. For simplicity, we assume that there are only two colors involed. The unfolding result is shown in Figure 3.10.

The neutral places p1 and p2 remain as two places. Place p3 is replaced by two places p31, p32 and place p4 by p41,p42 and place p5 by p51, p52. Since place p6 is associated only with one color, it remains as place p6.

The neutral transitions t1 and t6 remain as two transitions. Whereas, transition t2 is replaced by two transitions t21, t22 and transition t3 by t31, t32 and transition t4 by t41, t42. Transition t5 remains as t5 because only one color is associated with it.

The assignments of arcs are accomplished according to the corresponding input - output relationships in the colored net. For example, the arc connecting p1 to t1 remains as one arc, while the arc connecting p5 to t4 is replaced by two arcs connecting p51 to t41 and p52 to t42 respectively.

Firing rates are distributed to the corresponding transitions..

All the immediate transitions in the color net are in one group. Therefore, after the unfolding, all the immediate transitions are still in the same group. The firing probabilities are shown under the picture.

Finally, initial tokens are distributed as shown in Figure 3.10.

There are two ways to implement priority firings.

The first way is to modify the SOLUTION part. Refer to Figure 3.4. During the procedure of constructing the reachability tree, with each marking, SOLUTION goes to check every transition to see whether it is enabled. Now with priority firings, at the beginning, SOLUTION will ask user to input the priority levels of transitions. During the procedure of constructing the reachability tree, with every marking, SOLUTION first goes to check transitions which have the highest priority, if some of them are enabled, SOLUTION will not check the transitions with lower priorities. Instead, it finds out the new markings reached by the firing of those transitions with the highest priority. If none of the highest priority transitions are enabled, SOLUTION will check the transitions with the second highest priority . If some of them are enabled, it will find out the new markings reached. If none of them are enabled, SOLUTION will check the transitions with the next lower priority. This procedure is repeated over and over until the whole reachability tree is constructed.

The second way is to modify the USER FUNCTION module. Priority firings mean that when enabled, under certain conditions, a transition can not fire due to its lower priority. Here the word "condition" actually means markings. Hence priority firings may be viewed as marking dependent firings. When certain markings appear, the transition can not fire, the firing rate is 0.0. For the rest cases the firing rate is the specified rate associated with that transition for that color. Recall that in GSPNA , to deal with the marking dependent firing, the firing rate function is defined inside the USER FUNCTION module. Therefore, after the CGSPN is transformed into GSPN, priority firings can be defined inside the USER FUNCTION

according to the rules discussed above.

The time taken to execute the first method is less than that of the second. Because in building the reachability tree, by following the priority firing rules, the programme checks less transitions with the first method than with the second method. Presently, the programme adopts the second method. We will implement the first method in the near future.

As a matter of fact, conditions other than priority, e.g. inhibit arc, may be imposed on the transitions as well. Attention should be paid to ensure that the resulted net is still a Markov Chain.

The extended software package also has a friendly user interface. To use it, we suggest the following two steps:

1. Specification.

Type "descr". The user is guided to specify his model. He is asked to input places, transitions and the colors associated with them. The associated input and output places(colored or neutral) for each transition(colored or neutral) are also needed. When a transition in a certain color is declared to be timed, firing rate should be specified. Whereas, when it is declared to be immediate, probability of firing is required. Both of them could be marking dependent, in which case, the user has to define the external functions in the USER FUNCTION. The assignment of immediate transition groups as well as the initial tokens of different colors in each place is required.

Before quitting this procedure, the user should type the command "w" indicating that he wishes to save the file. Immediately, he is asked to give the name of the new file. Finally, type "q" to leave this procedure.

2. Solution

Type "gspn". The user is asked to input the name of the file containing the specification and the name of the output file which will contain output results. SOLUTION reads the file and begins to solve the model.

For all the marking dependent functions, the user should specify the color, the transition number and the marking dependent function inside the USER FUNCTION module.

The final numerical results are the probabilities of the number of different color tokens in each place. Further results could be derived from them.

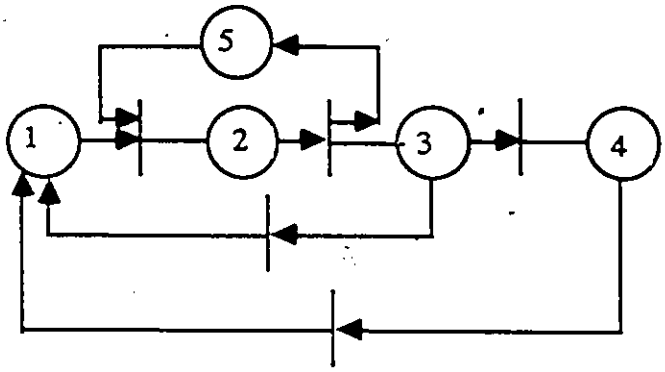


Figure 3.1 an example of PN

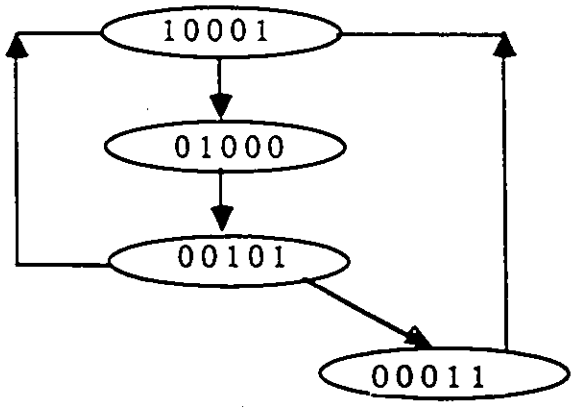


Figure 3.2 the reachability tree of the PN in Figure 3.1

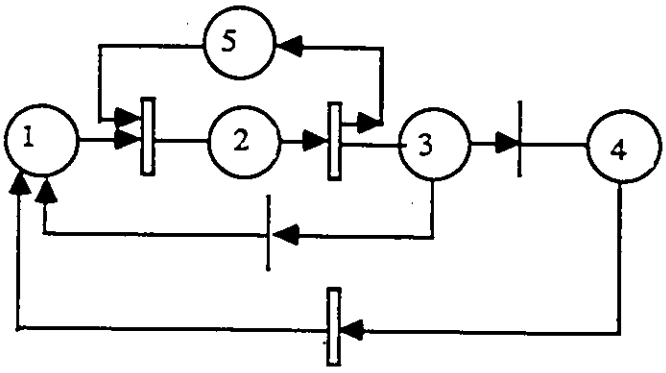


Figure 3.3 an example of GSPN

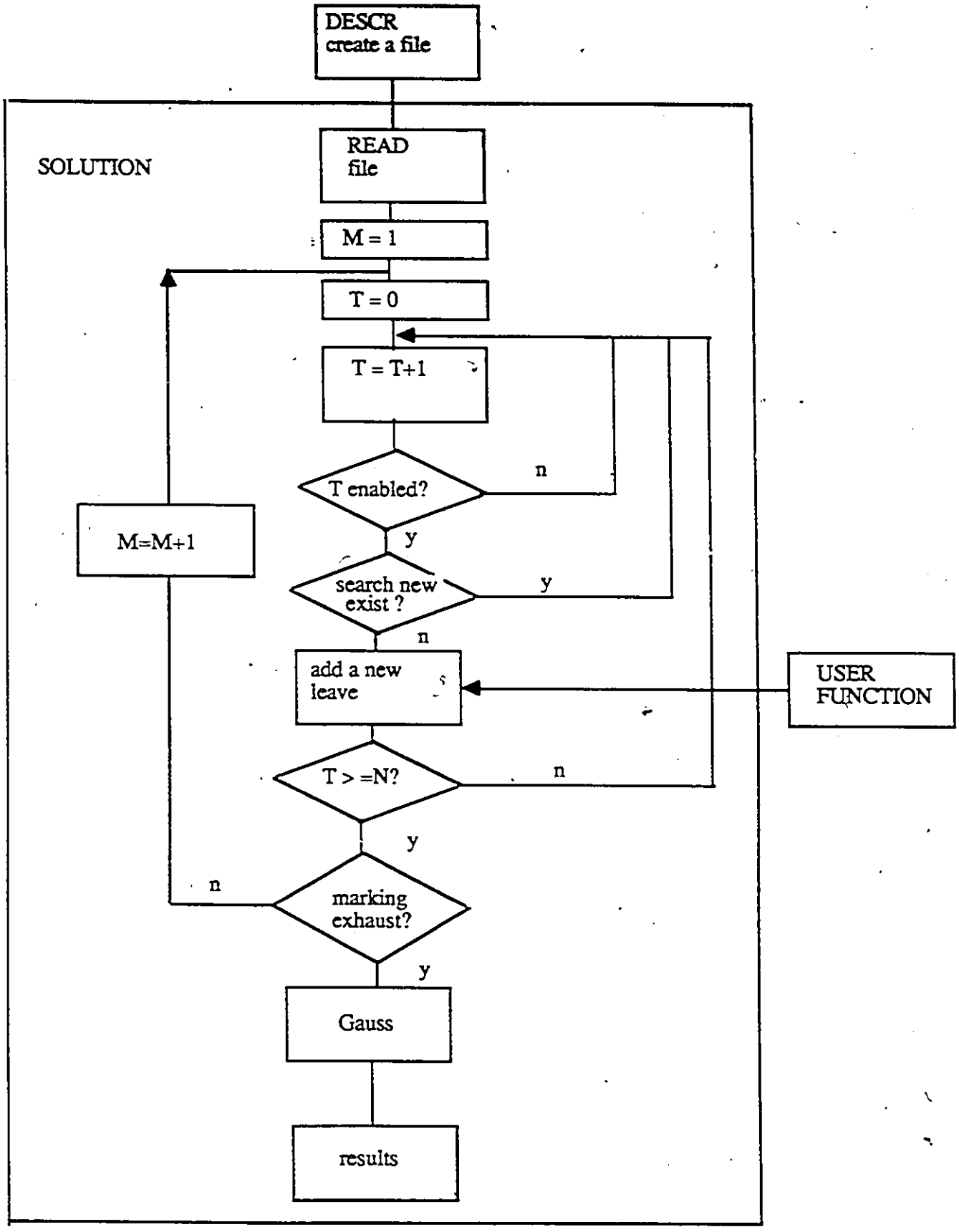


Figure 3.4 Block giagram of GSPNA

T: transition number.
M: marking number
N: number of timed transitions

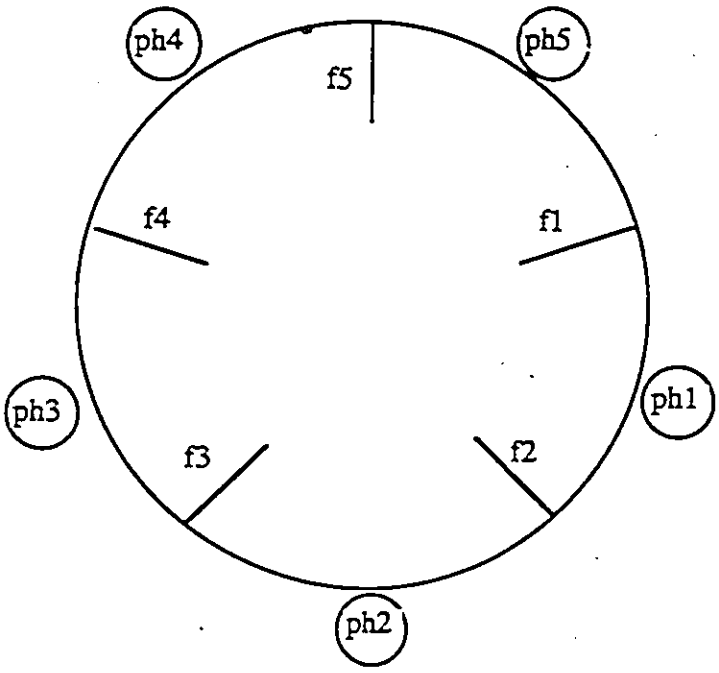


Figure 3.5 philosophers dining problem

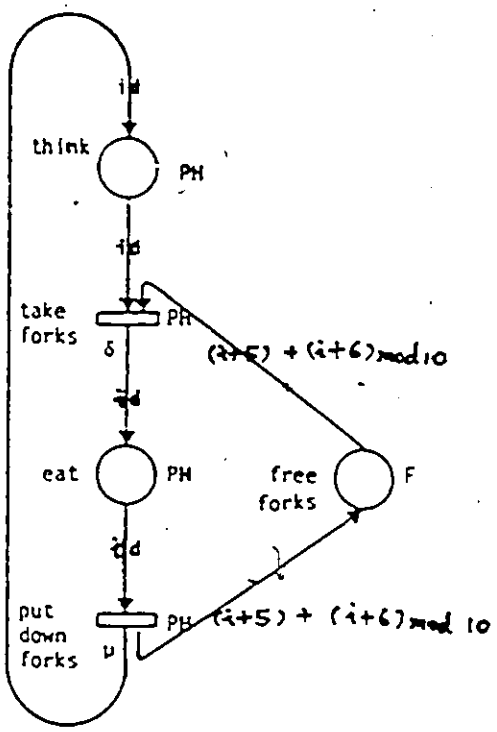


Figure 3.6 CSPN model of Figure 3.5

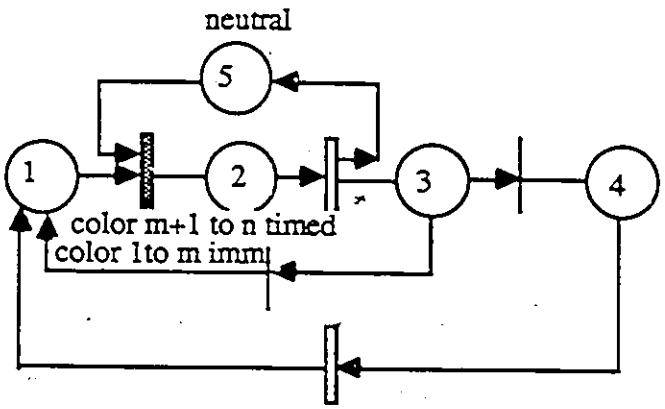
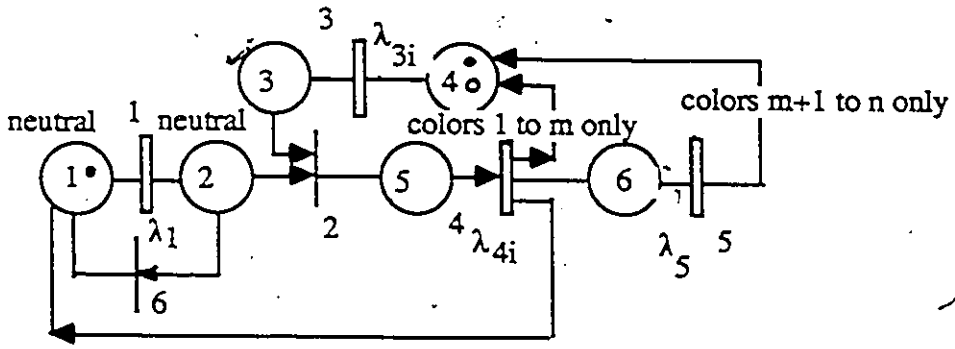


Figure 3.7 an example of CGSPN



$$p(6) = \begin{cases} 0 & \text{if transition 2 is enabled in any color} \\ 1 & \text{else} \end{cases}$$

$$p(2 \text{ in color } i) = \begin{cases} 1/k & \text{if transition 2 is enabled in k colors} \\ 1 & \text{else} \end{cases}$$

Figure 3.8 An example of CGSPN with neutral tokens.

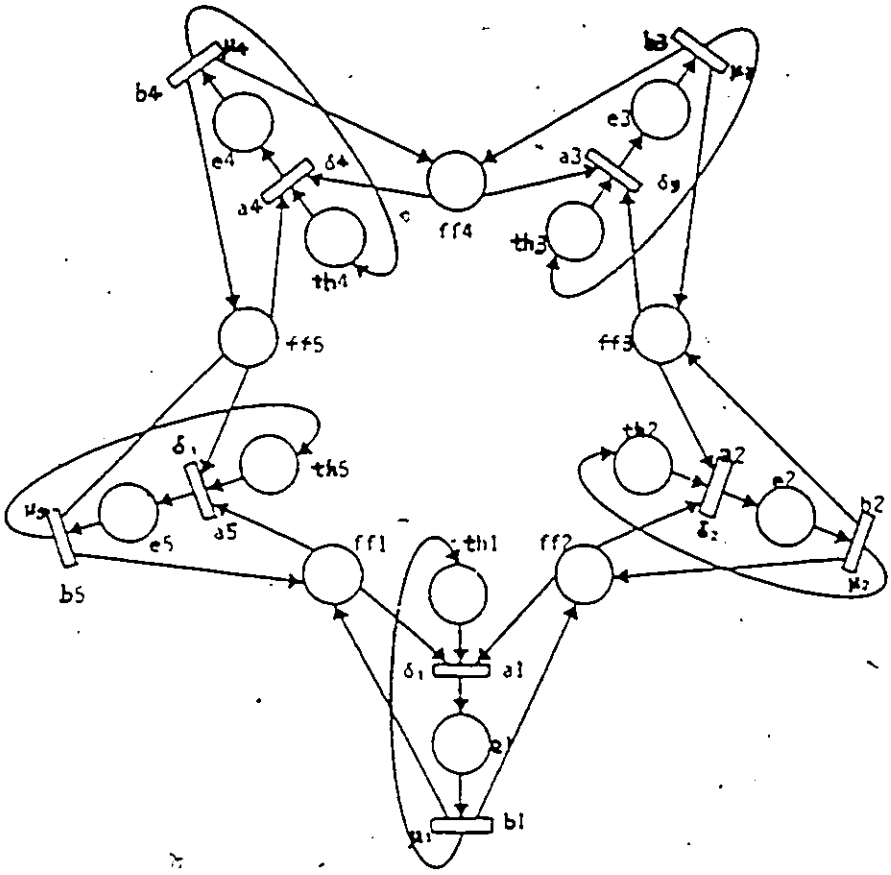
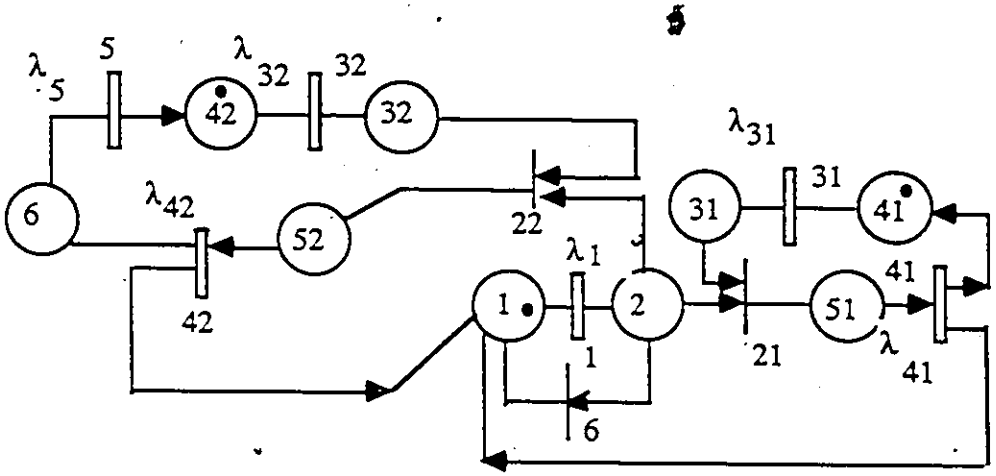


Figure 3.9 unfolding of the CSPN in Figure 3.6



$$p(6) = \begin{cases} 0 & \text{if either 21 or 22 is enabled} \\ 1 & \text{else} \end{cases}$$

$$p(2i) = \begin{cases} 1/2 & \text{if transition 2j } (j \neq i) \text{ is enabled} \\ 1 & \text{else} \end{cases}$$

Figure 3.10 The unfolding result of Figure 3.8

CHAPTER IV
An Integrated Data Voice System (IDVS)

This chapter describes a protocol of an Integrated Data Voice System (IDVS) for a LAN environment. [TEO 86]. Prototypes of the system were built in our lab. The performance of this system will be evaluated by our extended CGSPN.

4.1 Protocol Considerations.

In designing an integrated system for data and voice, attention should be paid to the differences between their performance requirements [TEO 86] [GRUB 83]. In integrated traffic applications, real time voice communications require both high throughput and low delay. In contrast, for data communications, interactive data services require low delay but allow low throughput whereas batch mode services require high throughput but allow high delay. Because a small percentage of lost packets is tolerable for voice, reliability may be sacrificed. However, reliability or information integrity is extremely critical for data traffic.

Figure 4.1 shows the configuration of the system and Figure 4.2 shows the functional layers of the IDVS. Each layer will be discussed separately in the following sections.

4.2 The physical and Data Link Layer

The physical layer and the medium access sublayer adopt the IEEE

802.3 Standard, which uses CSMA/CD as a medium access technique.

Most part of the data link layer is implemented in hardware (Intel 82586). The software of this layer only requires routines for initializing the LAN coprocessor, setting up command blocks for frame transmission, and preparing buffers for frame reception [TEO 86].

4.3 The Transport Layer

In the transport layer, different protocols were designed to handle data and voice seperately.

For data communication, the required protocol functions are:

connection establishment and termination (with error recovery);
data transfer (with error recovery, sequencing, duplication detection, segmentation and flow control).

For voice communication, the required protocol functions are:

call establishment and termination (with error recovery);
voice transfer (with packetization and reconstitution).

The transport layer is described by states. There are 6 states for voice and 4 states for data. They are:

0. --- Idle;
1. --- Demand response : waiting for destination to respond.
2. --- Data transfer.
3. --- Response pending : waiting for node's user process to respond.
4. --- Answer pending : waiting for end user to respond.
5. --- Demand answer : waiting for destination to answer.

The transport protocol commands and responses which are applicable for both data and voice communications are described as follows:

CR:	connection/call request
CC:	connection/call confirm
LR:	line request (for voice only)
DR:	disconnect request

The interface between the user process and the transport layer is of great interest in performance analysis, because it represents the coupling between these two layers.

The transport interface is described in terms of the transport primitives. For data and voice, the primitives are:

outgoing primitives

Connection request.
Disconnection request.
Connection accept.
Line request (voice only).

incoming Primitives

Connection indication.
Disconnection indication.
Connection established.
Line indication (voice only).

Outgoing primitives are issued by the user process and are passed to the transport layer. Whereas, incomming primitives are issued by the transport

layer and are passed to the user process.

The state transition from one state to another state is triggered by two sources, namely, outgoing transport primitives and packet commands received. Refer to table 4.1 and 4.2 for details.

The transport layer is implemented by three major modules. The first module provides functions to decode incoming transport service primitives, and takes appropriate actions such as updating transport states, forming packets and pushing them into the right queue. The second module decodes the packets received from the link layer and takes appropriate actions such as preparing outgoing transport service primitives. The last module is responsible for packet transmission and retransmission. The packets are fetched for command, voice and data queues according to their priority. The command packet has the highest priority, and the data packet has the lowest priority. Packet retransmission applies only on command or data packets.

4.4 The User Process Layer

The user process layer provides a connection between the user and the transport layer. Any information that an end user sends or receives uses this layer as a bridge for communication.

The user process interface between an end user and the user process is described in terms of the user process primitives. They are:

Connect
Disconnect
Data
Answer (voice only)

Issue of dial or type IDVS commands will activate a connect primitive; Issue of disconnect or hang up IDVS commands will activate a disconnect primitive; Issue of data or voice will activate a data primitive; Issue of answer IDVS command will activate an answer primitive. All of them will be sent to the user process.

The user process is implemented in terms of states. There are 3 states for data and 5 states for voice.

0. --- Close : idle.
1. --- Pending : waiting for connection acknowledge.
2. --- Open : virtual circuit available.
3. --- Line : waiting for line indication (voice only).
4. --- User : waiting for user to answer (voice only).

The state transition from one state to another state is triggered by two sources. i.e. from user process primitives and incoming transport primitives. Refer to table 4.3 and 4.4 for details.

The User process consists of two main modules. One of them processes the incoming transport service primitives and outputs, depending on the interactions, data or voice to the user terminal. The other module accepts data and voice (from the keyboard I/O and the voice I/O respectively) or commands them (from the IDVS command decoder), then processes and passes them to the transport layer.

An example of the establishment of a data virtual circuit connection is given in table 4.5 [TEO 85].

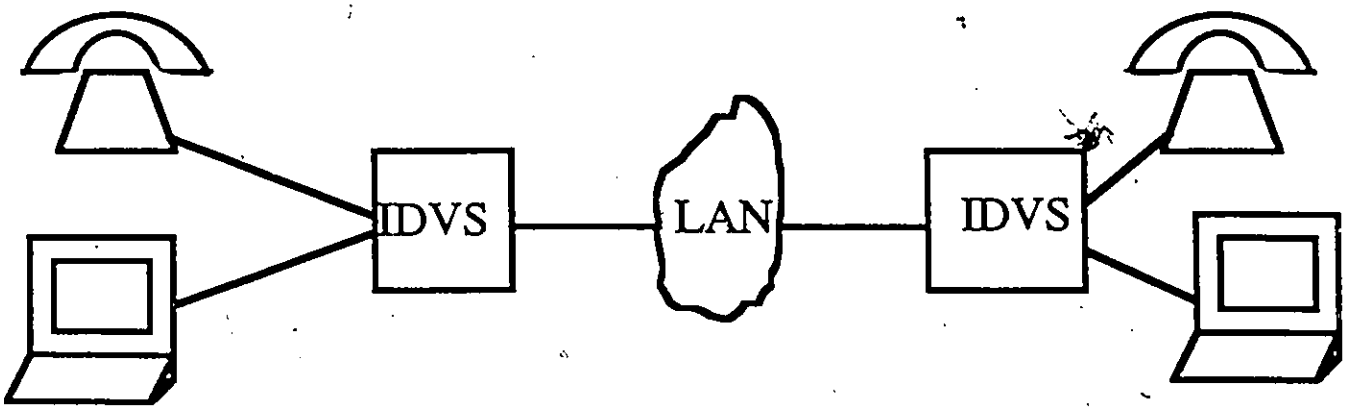


Fig.4.1 IDVS SYSTEM CONFIGURATION

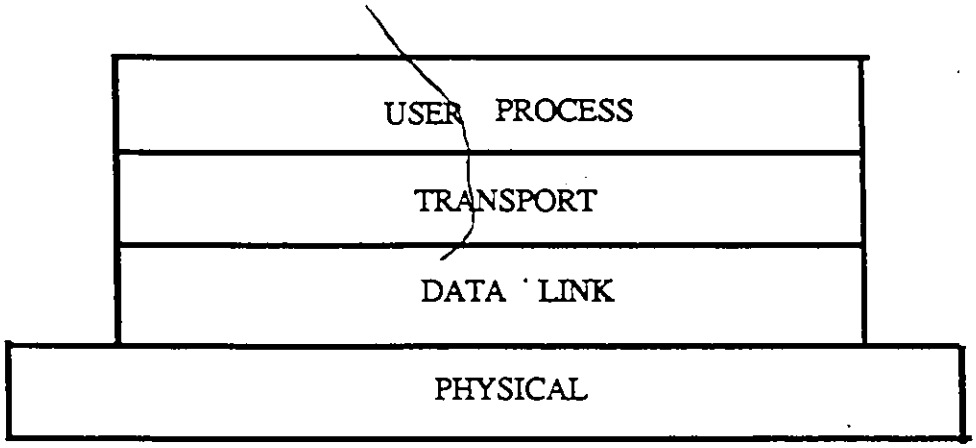


Fig.4.2 FUNCTIONAL LAYER OF THE IDVS

table 4.1 transport state transition caused by outgoing primitives

primit. / states	connection request	disconnection request	connection accept	line request
0 idle	send packet "CR" to state 1			
1 demand response				
2 data transfer		send packet "DR" to state 0		
3 response pending			send packet "CC" to state 4 (V) to state 2 (D)	
4 answer pending				send packet "LR" to state 2 (V)
5 demand answer				

table 4.2 transport state transition caused by packet command received

comm. / state	ÇR	CC	DR	LR
0 idle	"connection indication" to user proc. to state 3			
1 demand response		"connection established" to user proc. to state 5(V) to state 2(D)		
2 data transfer			"disconnect indication" to user proc. to state 0	
3 response pending				
4 answer pending				
5 demand answer				"line indication" to user proc. to state 2 (V)

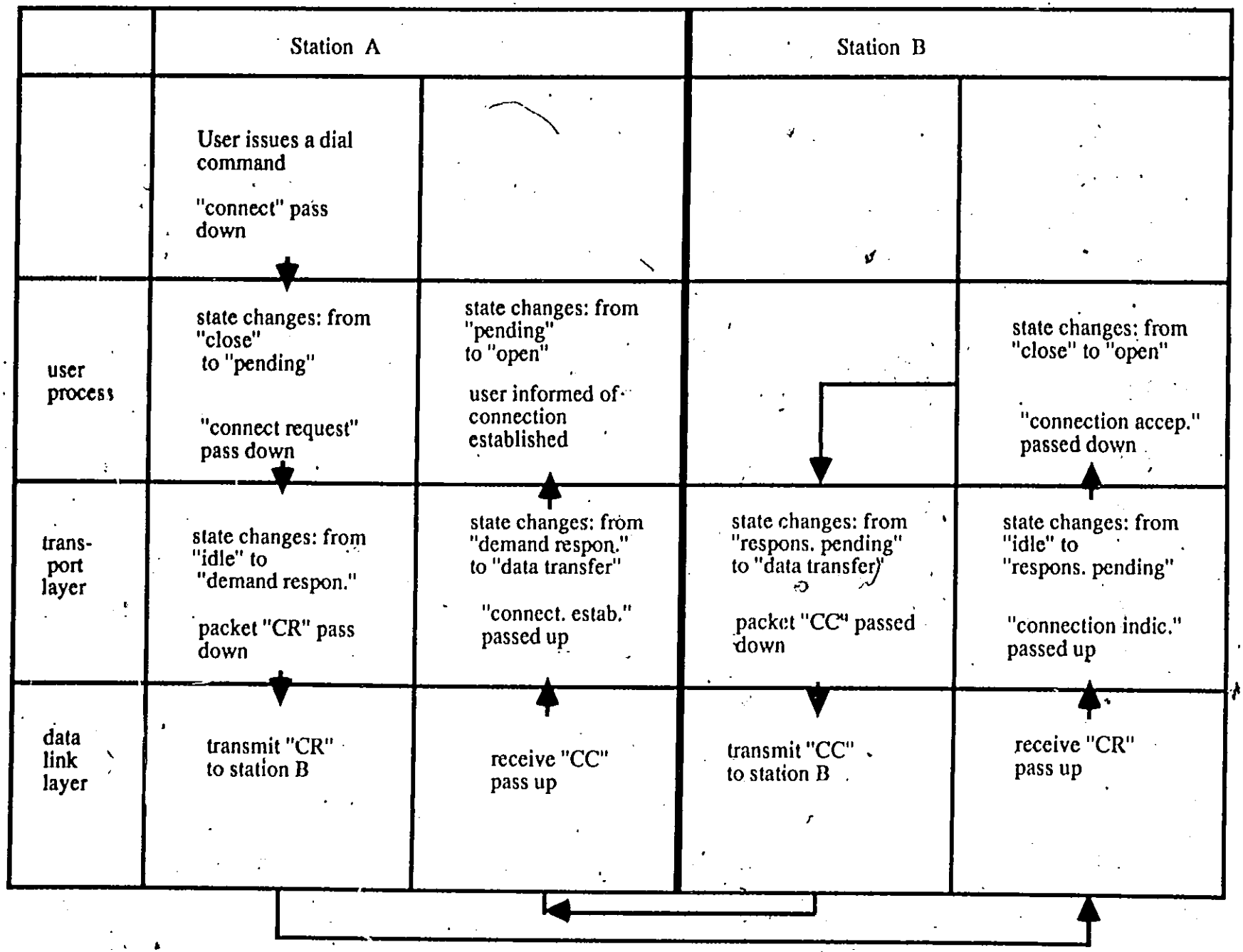
table 4.3 user process state transition caused by incomming transp. primitives

primitive / state	disconnect indication	connection established	connection indication	line indication
0 close			"connection accept" to transport user informed to state 4 (V) to state 2 (D)	
1 pending		connect message to user to state 3 (V) to state 2 (D)		
2 open	"Disconnect" to user to state 0			
3 line				answer message to user to state 2
4 user				

table 4.4 user process state transition caused by user process primitives

primitive / state	connect	disconnect	data	answer
0 close	"connection request" to transport to state 1			
1 pending				
2 open		"disconnect request" to transport to state 0	send data to transport	
3 line				
4 user				"line request" to transport to state 2

table 4.5 example of connection establishment



CHAPTER V
PERFORMANCE ANALYSIS OF THE INTEGRATED DATA VOICE SYSTEM BY EXTENDED CGSPN

In Chapter III, we discussed the CGSPN as a tool for multilayer protocol performance evaluation, and in Chapter IV, we described an Integrated Data Voice System (IDVS), particularly the protocols associated with it. In this chapter, we will build the performance model of the IDVS and analyze it.

5.1 Voice and Data Statistical Characteristics.

The dynamic behavior of a voice source has generally been modeled as a three state Markov Chain [KEKR 77] as shown in figure 5.1. The underlying assumption is that each call always starts and ends with a talkspurt. In general, a call consists of several talkspurts and silences with alternate occurrences. Experiments show that the average length of a talkspurt is 1.366 sec. and the average length of a silence period is 1.802 sec. [BRAD 65].

Next, we are going to establish voice and data models for later use. The information in a talkspurt is broken up into packets. Although by the method of stages we can present any kind of packet length distribution provided these distributions have rational Laplace transforms, this will complicate the model. Hence, we simply assume that the packet length distribution is exponential, and the number of voice packets N in a talkspurt period is geometrically distributed:

$$P(N=n) = p^{n-1}(1-p) \tag{5.1}$$

The total length T of the talkspurt is thus:

$$T = \sum_{i=1}^{N} X_i \tag{5.2}$$

where X_i is the length of each packet. All the X_is are random variables with independent, identical distributions. The density function is:

$$f(x) = \gamma e^{-\gamma t} \tag{5.3}$$

where $1/\gamma$ is the mean ppacket length.

Its Laplace transform is:

$$L_x(s) = \frac{\gamma}{(s+\gamma)} \tag{5.4}$$

The Laplace transform of the random variable T conditioned on N is:

$$L_{T|N}(s|n) = [L_x(s)]^n \tag{5.5}$$

By the theorem of total Laplace transform , we have:

$$\begin{aligned} L_T(s) &= \sum_{n=1}^{\infty} L_{T|N}(s|n)P(N=n) \\ &= \sum_{n=1}^{\infty} [L_x(s)]^n p^{n-1}(1-p) \\ &= \frac{(1-p)L_x(s)}{1-pL_x(s)} \\ &= \frac{(1-p)\gamma}{s+(1-p)\gamma} \end{aligned} \tag{5.6}$$

Hence, we see that under the above assumptions, the talkspurt length is exponentially distributed with parameter (1-p)γ. Since the mean talkspurt period is 1.366 sec, the γ and p we chose in modelling should satisfy the condition:

$$\frac{1}{(1-p)\gamma} = 1.366 \tag{5.7}$$

The silence period is also assumed to be exponential with μ:

$$\frac{1}{\mu} = 1.802 \tag{5.8}$$

where 1/μ is the mean silent period.

Now, let us look at data. We assume that data arrive in the form of messages. The intrerarrival time distribution between messages is exponential. A message is broken up into packets at the Transport layer. If we assume that the packet length distribution is exponential and the number of packets in one message is geometrically distributed, then in the same way as voice, we know that the message length should be exponential. If the interarrival time is analogous to the silence period of voice, and the message length to talkspurt period, we see the characteristics of both voice and data are similar, the difference is the statistical parameters defining each process.

In summary, the voice model we will use is the following:

1. The voice source is modeled as a three state Markov Chain.
2. A talk is composed of several talkspurt and silence periods with alternate occurrence.
3. The talkspurt and silent length distributions are exponentional.
4. The information in a talkspurt period is broken up into packets of exponentially distributed length, the number of packets satisfying

a geometric distribution.

The data model we will assume is:

1. Data arrive in the form of messages.
2. The interarrival time and message length distributions are exponential.
3. The messages are broken up into packets of exponentially distributed length, the number of packets satisfying a geometric distribution.

The behavior of data fits the exponential distribution assumption well, while the behavior of voice, especially the silent period, is approximated less well by the exponential assumption. However, for the purpose of simplifying the model, we have to make the above assumptions.

5.2 The CGSPN Model of the System

The CGSPN model of the IDVS we built is shown in Figure 5.2. It consists of 34 places, 23 transitions and two types of colors, one representing voice and another representing data. The meanings associated with each place and transition are summarized in table 5.1.

Although we may represent every detail of the system, this may result in too large a net to be solved. In addition, we are interested in the global behavior of the multilayer protocol. Therefore, some simplifications are allowed in order to represent only the signigicant factors. We assume that the sending and receiving are two separate processes in one station. Hence, in the model, one side is only sending messages while the other side is only receiving them. To avoid representing the details of Data Link layer and

Physical layer, we model them simply as an exponential delay experienced by each packet. We further assume that all the transition errors in data packets are recovered at the Transport layer, and since voice can tolerate some errors, retransmission is not required for both data and voice.

Recall that the User process layer and the Transport layer are described by states, and the interfaces between layers are described by primitives. The places in the model, except 1,14,15,27,28, are corresponding to the states of each layer, the timed transitions except 12,19-23, are corresponding to the receiving, processing and transmitting of primitives and packet commands. We note that data and voice may have different routings.

There are two neutral places, place 33 and place 34. Neutral tokens in the two places represent the two CPUs executing the protocol in the sending and receiving side respectively. The transitions in the sending side, which represent the starting of the sending CPU operation, have neutral place 33 as one of their input places. The transitions in the sending side, which represent the ending of the sending CPU operation, have neutral place 33 as one of their output places. In the same way, the transitions in the receiving side, which represent the starting of the receiving CPU operation, have neutral place 34 as one of their input places. The transitions in the receiving side, which represent the ending of the receiving CPU operation, have neurtral place 34 as one of their output places.

Transitions 19-23 together with places 27,29-32 represent the exponential delay in the Data Link and the Physical layer experienced by each packet.

One important feature of the Transport layer is the flow control. It is

used to regulate the amount of flow between stations so as to avoid buffer overflow.

For data transmission in this model, the flow control is realized by a sliding window of size n, where n is integer. Since we assumed that no retransmission is required, the essence of the sliding window is that the sender is not allowed to send packet i+n before i has been acknowledged. This flow control strategy is represented by place 28. Initially, there are n tokens in place 28, indicating that the maximum number of packets the sender may send without acknowledgement is n. Whenever one packet is sent, the number of tokens in place 28 decreases by 1. Whenever one packet is received, the receiver will send an acknowledgement back through place 27 and transition 19. Hence, the number of tokens in place 28 increases by 1, meaning that the sender may send one more packet.

At the first glance, it seems that flow control for voice is not necessary. But, to avoid traffic congestion caused by the flooding of voice packets, a sliding window control is also used in the model. Actually, over a local area network, the delay caused by waiting for an acknowledgement is not significant if we select the window size appropriately. We emphasize that the window size for data is not necessary the same as that for voice. They are represented by tokens of different color respectively.

The statistics of data and voice are represented by transitions 8-12. The firing of transition 8 denotes the processing and transmitting of one packet. Then a token is put into place 14. At this time instant, with probability p, transition 11 fires. If there is at least one token in place 28, transition 8 is enabled again indicating that another packet is ready to be processed or transmitted. Or, with probability q, transition 10 fires, and one token is put into place 15 representing the silence period for voice or the message

interarrival time for data. Or, with probability r, transition 9 fires, and the current connection ends. The firing rate γ of transition 8 and the firing probability p should satisfy (5.7) for voice, and

$$p + q + r = 1 \tag{5.9}$$

for transitions 9, 10, 11.

The firing rates and the probabilities of firing of the transitions are listed in table 5.2.

In order to reduce the traffic on the network, the silent period of voice is not transmitted. A buffer delay is added to the received packets before delivering to the destination user. In this way, the silent period is used to transmit data packets if there are any. We notice that, as described in Chapter IV, voice packets are assigned higher priority than data packets, and command packets have the highest priority. Hence, in the CGSPN model, the transitions in the sending and receiving part are assigned with priorities of three levels:

1. Transitions 2-7 in the sending part and transitions 13-17 in the receiving part may fire whenever they are enabled by any tokens, voice or data. They represent the service for primitives and command packets. Note, here that data and voice commands have the same priority.
2. If transition 8 is enabled by voice, it may fire only when transitions 2-7 are not enabled by either voice or data. In the same way, if transition 18 is enabled by voice, it may fire only when transitions 13-17 are not enabled by either voice or data.These two transitions represent the service for voice packets

3. If transition 8 is enabled by data, it may fire only when transitions 2-8 are not enabled by voice, and transitions 2-7 are not enabled by data. In the same way, if transition 18 is enabled by data, it may fire only when transitons 13-18 are not enabled by voice,and transitions 13-17 are not enabled by data.

The rest of the transitions are independent of sending and receiving parts. Therefore, they may fire whenever they are enabled. They could be viewed as having the same priority as command packets in the model.

5.3 Numerical Results and Discussions

The model presented in section 5.2 is analyzed by CGSPNA. The mean length of a voice connection (not includes the connection establishment) is 2.5 min. and the mean length of a data connection (not includes the connection establishment) is 1.7 min.

For the purpose of comparison, we analyze three cases:

1. The whole system is used exclusively for voice connections.
2. The whole system is used exclusively for data connections.
3. The whole system is used for integrated data voice connections.

For convenient reference, we define the token representing voice as voice token, and the token representing data as data token.

A token, voice or data, in place one indicates that the CPU is ready to accomodate another connection of the same type. Hence, the probability P_0 of no token in place one is the probability that the sending CPU is busy and cannot accept new connections of that type. In this model, connections arrive

to the system at a Poisson rate λ and are lost if the system is not available. Therefore, the actual throughput of the system is $\lambda(1-P_0)$.

Varying the firing rate λ_v of transition one for voice token, we get the throughput of voice versus the offered voice load, which is shown in Figure 5.3. In the same way, varying the firing rate λ_d of transition one for data token, we get the throughput of data versus the offered data load, which is shown in Figure 5.4.

The results in Figure 5.3 show that the throughput for voice in case three is slightly less than that in case one. The performance of voice is nearly unaffected due to the fact that voice packets are assigned higher priority than data packets. The slight difference is caused by the fact that command packets have higher priority than voice packets and that the CPU service for any packets is non-preemptive. There are chances that voice packets are not served until the service for command packets is finished or until the service for a data packet, which was already in service when the voice packets arrive, is finished.

In contrast, the result in Figure 5.4 show that the throughput for data in case three is much less than that in case two. This is expected, since data packets have the lowest priority.

The system is said to be busy serving voice (or data) during the whole period of a voice (or data) connection. Let {B} denote the event that the integrated system is busy. {B} refers to that the system is busy serving a voice connection, or a data connection, or both of them at the same time. we are going to obtain the probability P(B). We have:

$$P(B) = P(B_v + B_d) = P(B_v) + P(B_d) - P(B_v B_d)$$

where:

B_v : the system is busy serving voice;

B_d : the system is busy serving data;

B_v and B_d are not mutually exclusive.

Obviously, $P(B_v)$ is the probability that no voice token is in place one, $P(B_d)$ is the probability that no data token is in place one, and $P(B_v B_d)$ is the probability that neither voice nor data token is in place one. The results obtained for $P(B_v)$, $P(B_d)$, $P(B)$ versus offered voice load and data load are shown in Figure 5.5 and Figure 5.6 respectively.

Natually, when the offered voice load increases , the system will spend more time serving the connections. Hence, $P(B_v)$ increases. We also notice that, in Figure 5.5, $P(B_d)$ also increases as voice load grows. The reason is that with the increasing of the offered voice load, the delay experienced by each data connection grows due to the lower priority of data packets. Hence the average time the system busy serving data connection increases, and so does $P(B_d)$.

The results in Figure 5.6 show that $P(B_d)$ increases with the growing of the offered data load. However, regardless of the variation of data load, $P(B_v)$ is almost flat due to the higher priority of voice packets.

Naturally, P(B) increases when either voice or data load increases.

One of the points we want to show in this chapter is that by correctly identifying the meaning of the states in the Markov Chain associated with the CGSPN model, we are able to use the CGSPN to obtain performance indices of interest in many complex systems.

Table 5.1 Legend for the CGSPN in Figure 5.1

TRANSITIONS

1	arriving of commands for establishing another connection.
2	end of processing and "connect" primitive and passing down "connection request" primitive
3	end of receiving "connection established" primitive
4	end of receiving "line indication" primitive
5	end of receiving "connection request" primitive and transiting "CR" packet
6	end of receiving "CC" command packet and passing up the "connection established" primitive
7	end of receiving "LR" command packet and passing up the "line indication" primitive(voice only).
8	end of transmitting one packet
9	connection ends
10	beginning of idle(voice only) | beginning of message interarrival(data only)
11	another packet is ready for transmission
12	end of idle (voice only) | Arriving of another message (data only)
13	end of receiving and processing "CR" and passing up "connection indication" primitive
14	end of receiving "connection accept" primitive and transmitting "CC" packet
15	end of receiving "line request" primitive and transimitting "LR" command packet (voice only)
16	end of processing and receiving "connection indication" packet and passing down "connection accept" primitive
17	end of receiving "answer" primitive and passing down "line request" primitive
18	end of receiving one packet and sending the acknowledgement
19	end of transmitting acknowledgement through DL
20	end of transmitting one packet through DL
21	end of transmitting "LR" packet through DL
22	end of transmitting "CC" packet through DL
23	end of transmitting "CR" packet through DL

PLACES

1	ready to accomodate another connection
2	"close" state of UL at SS
3	"pending" state of UL at SS
4	receiving "connection established" primitive at SS
5	"line" state of UL at SS
6	"open" state of UL at SS
7	receive "line indication"
8	"idle" state of TL at SS
9	"demand response" state of TL at SS
10	receiving "CC" command packet at SS
11	"demand answer" state of TL at SS (voice only)
12	receiving "LR" command packet at SS (voice only)
13	"data transfer" state of TL at SS
14	finishing transmitting one packet(voice or data)
15	idle period (voice only), inter-message-arrival time(data only)
16	"idle" state of TL at RS
17	receive "CR" command packet
18	"response pending" state of TL at RS
19	receive "connection accept" primitive
20	"answer pending" state of TL at RS
21	receive "line request"
22	"data transfer" state of TL at RS
23	receiving packet
24	"close" state of UL at RS
25	"user" state of UL at RS
26	"open" state of TL at RS
27	acknowledgement is sent through DL
28	flow control
29	packet is transmitted through DL
30	packet "LR" is transmitted through DL
31	packet "CC" is transmitted through DL
32	packet "CR" is transmitted through DL
33	CPU executing the protocol at SS
34	CPU executing the protocol at RS

DL:	Data Link Layer
TL:	Transport Layer
UL:	User Process Layer
SS:	Sending Side
RS:	Receiving Side

Table 5.2 firing rates and probabilities of firing for the the transitions in the CGSPN of Figure 5.1

	color 1(voice)	color 2(data)
1	λ_v	λ_d
2	15	15
3	20	20
4	20	---
5	15	15
6	15	15
7	15	---
8	3.7	6.25
9	0.004	0.004
10	0.196	0.496
11	0.8	0.5
12	0.6	2
13	15	15
14	15	15
15	15	---
16	15	15
17	20	---
18	4	6.5
19	50	50
20	50	50
21	50	---
22	50	50
23	50	50

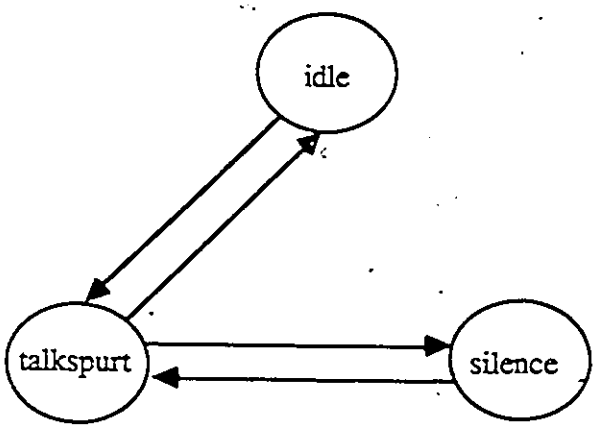


Figure 5.1 Three State Markov Chain Model of a Voice Call

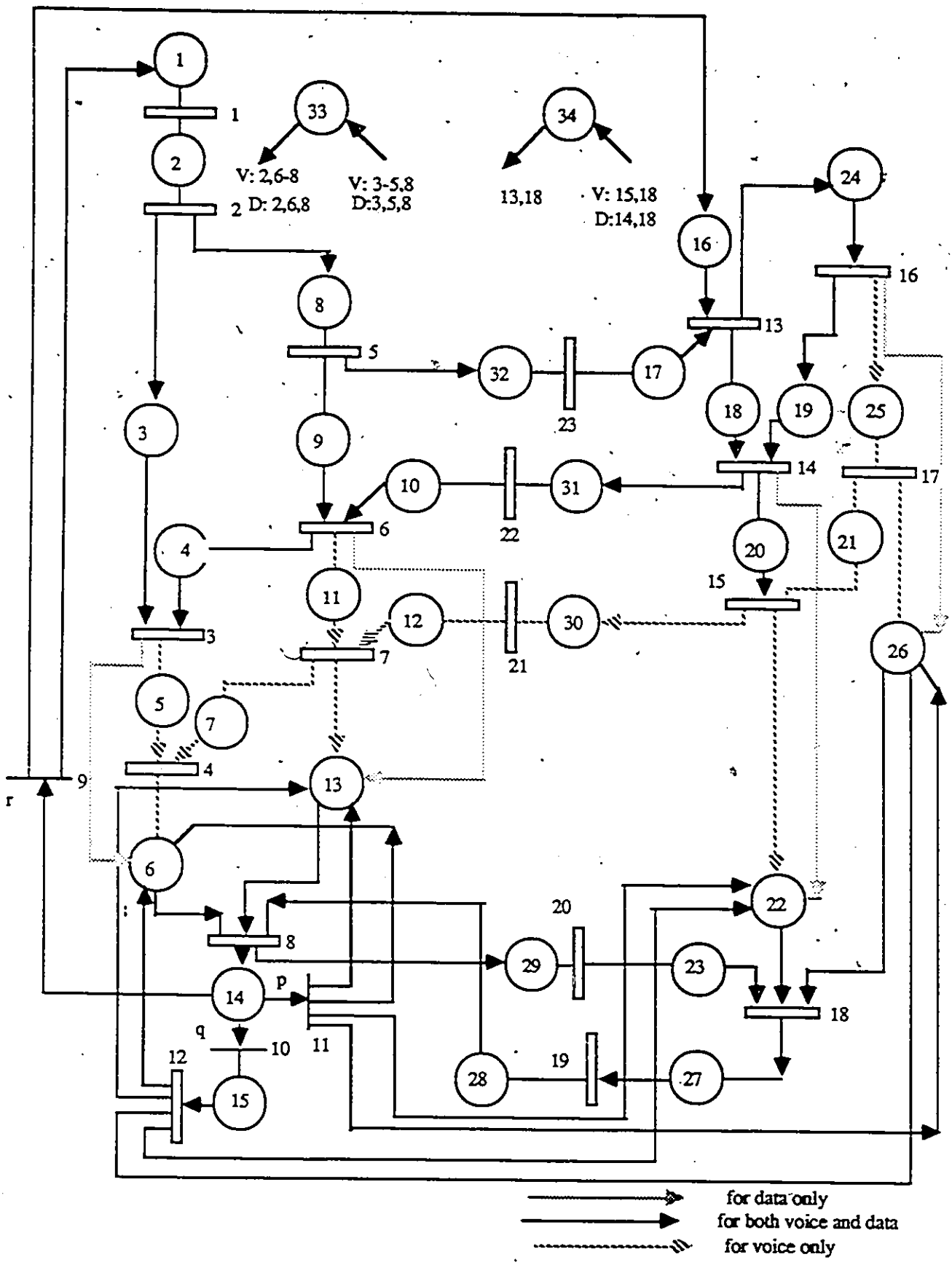


Figure 5.2 CGSPN model of the Integrated Data Voice System

Figure 5.3 Voice throughput versus offered voice load

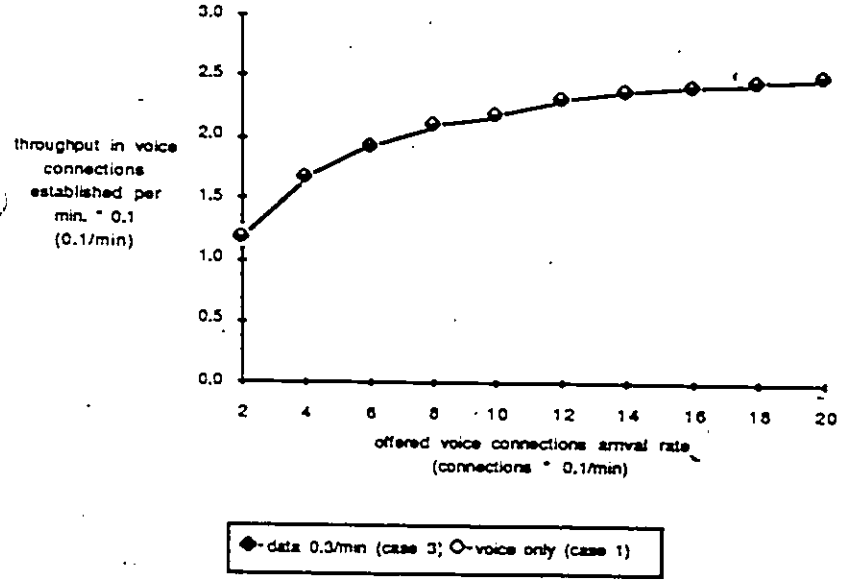


Figure 5.4 Data throughput versus offoderd data load

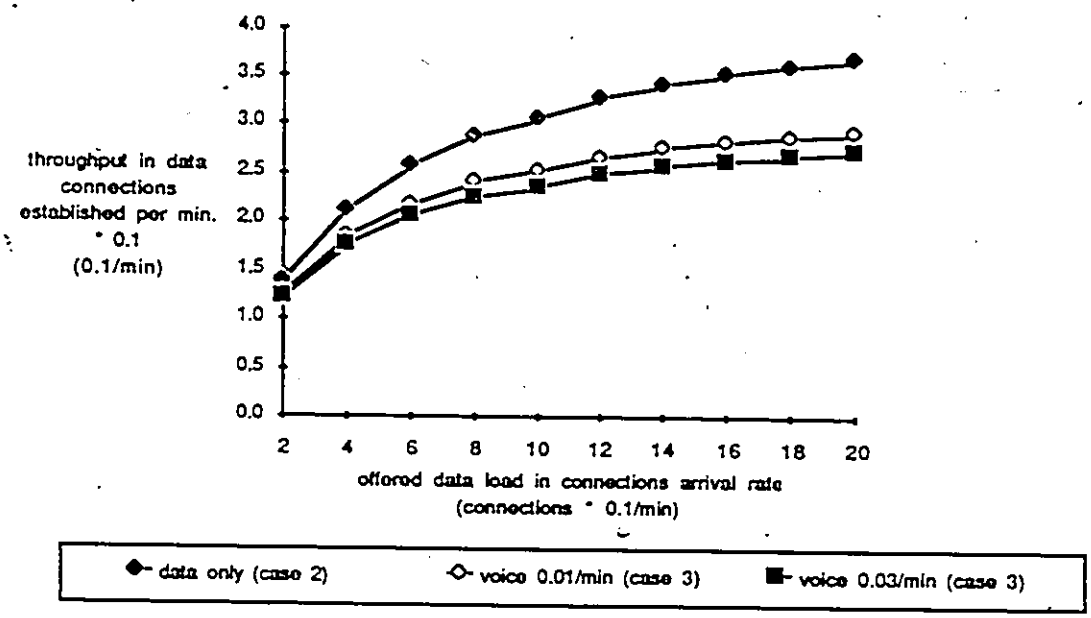


Figure 5.5 Probability of system busy versus offered voice load (case 3)

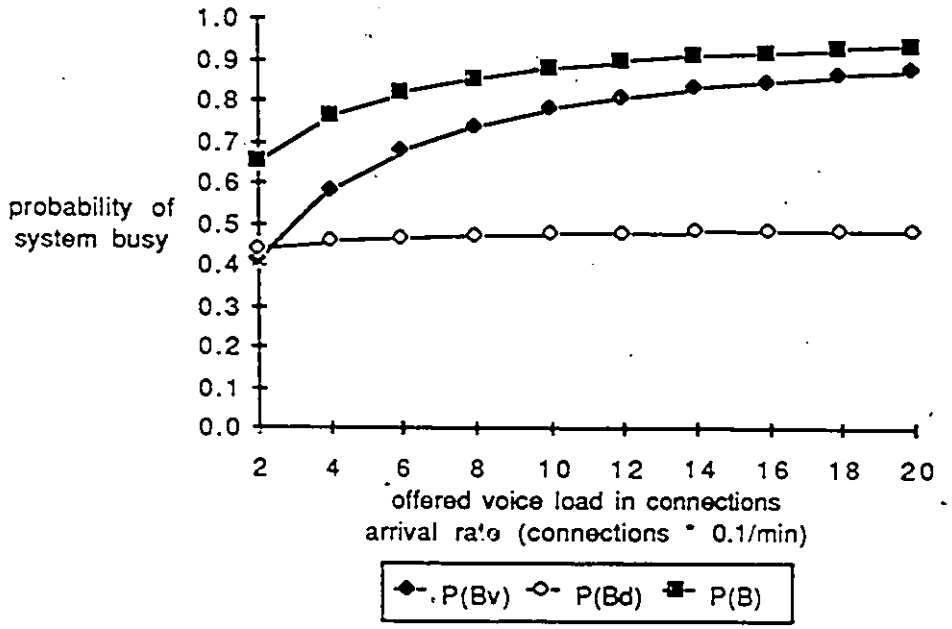
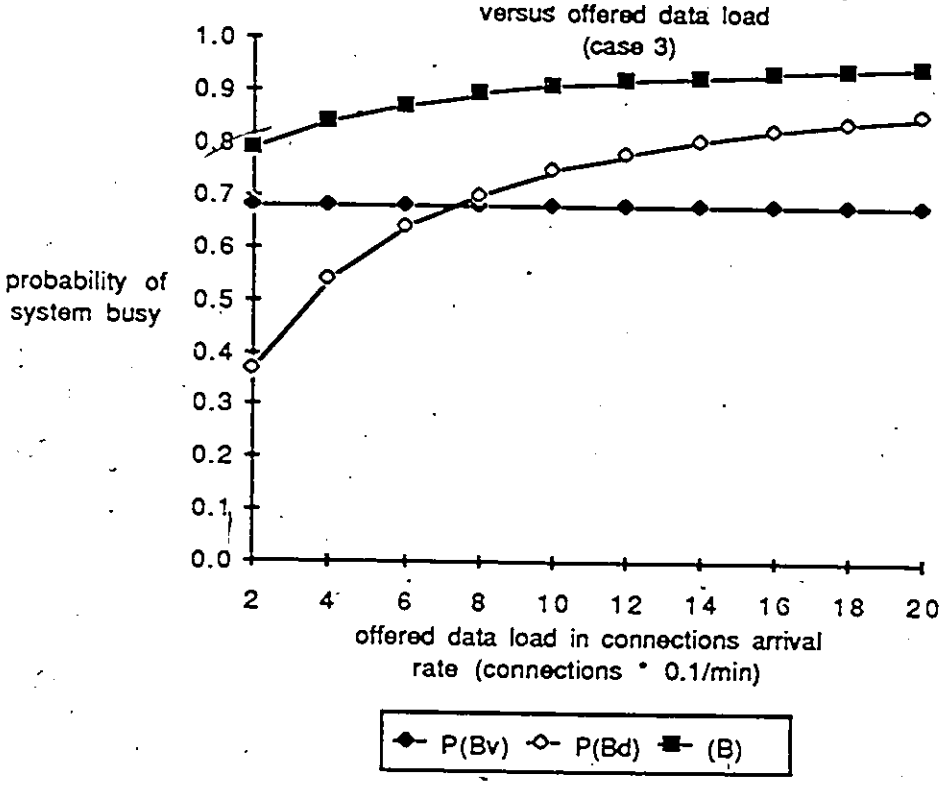


Figure 5.6 Probability of system busy versus offered data load (case 3)



CHAPTER VI
CONCLUSIONS

In this thesis, we proposed a tool for integrated system multilayer protocol performance evaluation, namely, Colored Generalized Stochastic Petri Nets (CGSPN). We extended the CGSPN by introducing the new concepts of "neutral tokens" and "priority firings". Problems raised by the extentions were solved by the theory of stochastic processes. It was shown that the extended CGSPN is still a Markov Stochastic process. In order to obtain the numerical results from the extended CGSPN, we built a PASCAL program to extend the GSPNA, which is a software tool for analyzing GSPN. Steps to apply the extended CGSPN to integrated system protocol performance modelling and to solve the model by the extended software package were examined. Finally, as an application, an Integrated Data Voice System, particularly the protocol concerning it, was described, and the performance of the multilayer protocol was evaluated by the extended CGSPN.

The proposed extended CGSPN was shown to be suitable for integrated system multilayer protocol performance evaluation in that it is able to model and analyze synchronization, concurrency, cooperation, resource contention, class identification as well as priority services, which are difficult to be analyzed by either Finite State Machines or Queueing networks.

In addition, it is worth mentioning that besides multilayer protocol, CGSPN can also be applied to many other performance evaluation cases. For example, the performance of the VME bus arbitration mechanism, which

decides who may use the data bus according to the priority levels as well as certain rules.

Although the extended CGSPN is a promising tool for modelling complex protocols, one weakness hampered its applications. The underlying Markov Chain tends to be too large to be managed. Hence, there is a need to find straight forward solutions for PNs, which are just like the product form solutions for queueing networks. Since there are many philosophical differences between PNs and Queueing networks, the simple solution for PNs will not necessarily be a product form. We feel that it is an extremely interesting topic to find simple solutions for PNs.

- This thesis has contributed in:

1. extending the class of Petri Nets;
2. extending the GSPNA software package;
3. presenting a tool for integrated multilayer protocol performance evaluation.
4. modelling and analyzing the protocol performance of an Integrated Data Voice System.

We hope that this work will stimulate the interest of the researchers in the area of integrated networks, as well as those researchers in Petri Nets.

APPENDIX A

In Generalized Stochastic Petri Nets, when more than one timed transitions are enabled, the transition which is associated with the shortest delay will fire first. Since the delays are random variables with exponential distributions, every transition has the chance to fire first. We are going to find the probability of firing transition i first.

We know that for a stochastic process M_t, if the interarrival time is exponentially distributed, then the arriving process is Poisson process [PAPO 84]. The Poisson process distribution is

$$P(N_t = k) = \frac{(\lambda t)^k}{k!} e^{-\lambda x} \tag{1}$$

where

N_t: the number of arrivals during period t;

λ: the arriving rate.

Let X represent the interarival time, then the density distribution of X is

$$f(x) = \lambda e^{-\lambda x} \tag{2}$$

The distribution is

$$F(x) = P(X \leq x) = 1 - e^{-\lambda x} \tag{3}$$

Now, consider the merging of two independent Poisson processes, say

M_{1t}, M_{2t}, with arriving rates λ_1,λ_2. Let X_{i1}, i=1,2, denote the time period from zero to the first arrival of process i. The probability that the first arrival is from M_{1t} can be obtained according to the law of total probability [PAPO 84]

$$P[X_{11} < X_{21}] = \int_0^{+\infty} P[X_{21} > x | X_{11} = x] f_1(x) dx$$
$$= \int_0^{+\infty} P[X_{21} > x] f_1(x) dx \tag{4}$$

where $f_1(x)$ is the density distribution function of the interarrival time of process M_{1t}.

$\{ X_{21} > x\}$ means that there are none arrivals from M_{2t} in the period from the beginning to time x, hence according to (1),

$$P[X_{21} > x] = P(N_x = 0) = e^{-\lambda_2 x} \tag{5}$$

$$P[X_{11} < X_{21}] = \int_0^{+\infty} e^{-\lambda_2 x} \lambda_1 e^{-\lambda_1 x} dx$$
$$= \frac{\lambda_1}{\lambda_1 + \lambda_2} \tag{6}$$

Next, consider the merging of n independent Poisson processes, M_{1t}, M_{2t}, ...M_{nt}, with arriving rates λ_1, λ_2, ...λ_n respectively, let X_{i1}, i=1...n, denote the time period from zero to the first arrival of process i. Then they are independent, hence:

$$P[min(X_{11}, X_{21}, ...X_{n1} \le t] = 1 - P[min(X_{11}, X_{21}, ...X_{n1} > t]$$
$$= 1 - P(X_{11} > t)P(X_{21} > t)...P(X_{n1} > t) \tag{7}$$

$$P(X_{i1} > t) = 1 - P[X_{i1} \le t] = e^{-\lambda_i t} \tag{8}$$

$$P[min(X_{11}, X_{21}, \ldots X_{n1}) \leq t] = 1 - e^{-\sum_{i=1}^{n} \lambda_i t} \quad (9)$$

Hence, $\{\min(X_{11}, X_{21}, \ldots X_{n1}) \leq t\}$ is also exponentially distributed with parameter $(\lambda_1+\lambda_2+\ldots+\lambda_n)$.

With the merging of n Poisson process, the event that the first arrival is from process i is$\{X_{i1} < \min(X_{j1}, j \neq i)\}$. Since we have shown that both X_{i1} and $\{\min(X_{j1}, j \neq i)$ are exponentially distributed with parameter λ_i and $(\lambda_1+\lambda_2+\ldots+\lambda_{i-1}+\lambda_{i+1}+\ldots+\lambda_n)$ respectively, by (6) we have:

$$P[X_{i1} < min(X_{j1}, j \neq i)] = \frac{\lambda_i}{\sum_j \lambda_j} \quad (10)$$

With the problem we proposed at the beginning of the appendix, if we look at the transition firing as an arrival, then transition t_i fires first means that it arrives first, hence (3.9) is the probability that t_i fires.

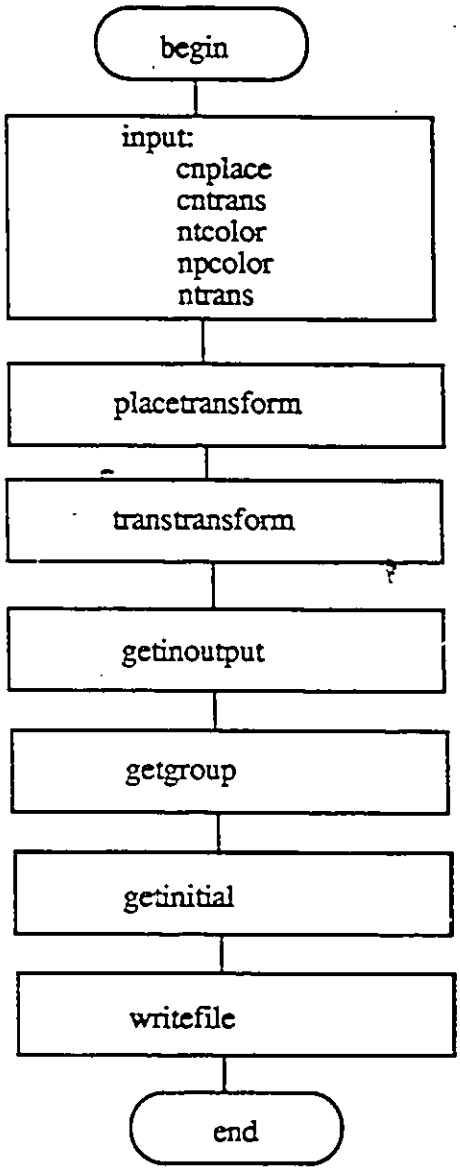
In CGSPN without priority firings, the denominator of (10) should consist of all the transitions which are enabled by a certain marking. In CGSPN with priority firings, the denominator of (10) should consist of all the transitions which are enabled by a certain marking and are allowed to fire according to the priority principles. Hence, (3.19) and (3.20) are justified.

APPENDIX B

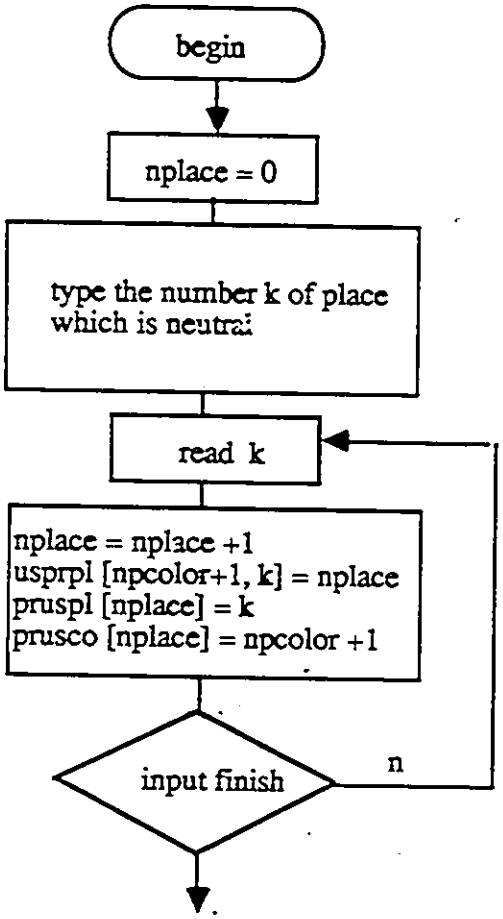
B.1. List of Terms and Variables.

nplace	total number of places in GSPN.
ntrans	total number of trans. in GSPN.
cnplace	total number of places in CGSPN.
cntrans	total number of trans. in CGSPN.
ntcolor	total number of colors of trans.
npcolor	total number of colors of places.
m0 [i]	initial number of tokens in place i in GSPN.
group [i]	the number of group transition i belongs to in GSPN.
inp [i,j]	the multiplicity of input arcs of trans. i from place j.
out [i,j]	the multiplicity of output arcs of trans. i to place j.
usprpl [i,j]	the corresponding place in GSPN of the place j of color i in CGSPN.
pruspl [i]	the corresponding place in CGSPN of the place i in GSPN.
prusco[i]	the corresponding color in CGSPN of the place i in GSPN.
usprtr [i,j]	the corresponding trans. in GSPN of the trans. j of color i in CGSPN.
prustr [i]	the corresponding trans. in CGSPN of the trans. i in GSPN.
prustco[i]	the corresponding color in CGSPN of the trans. i in GSPN.
rateprob[i]	the firing rate or probability of finring of transition i in GSPN

MAIN PROGRAM

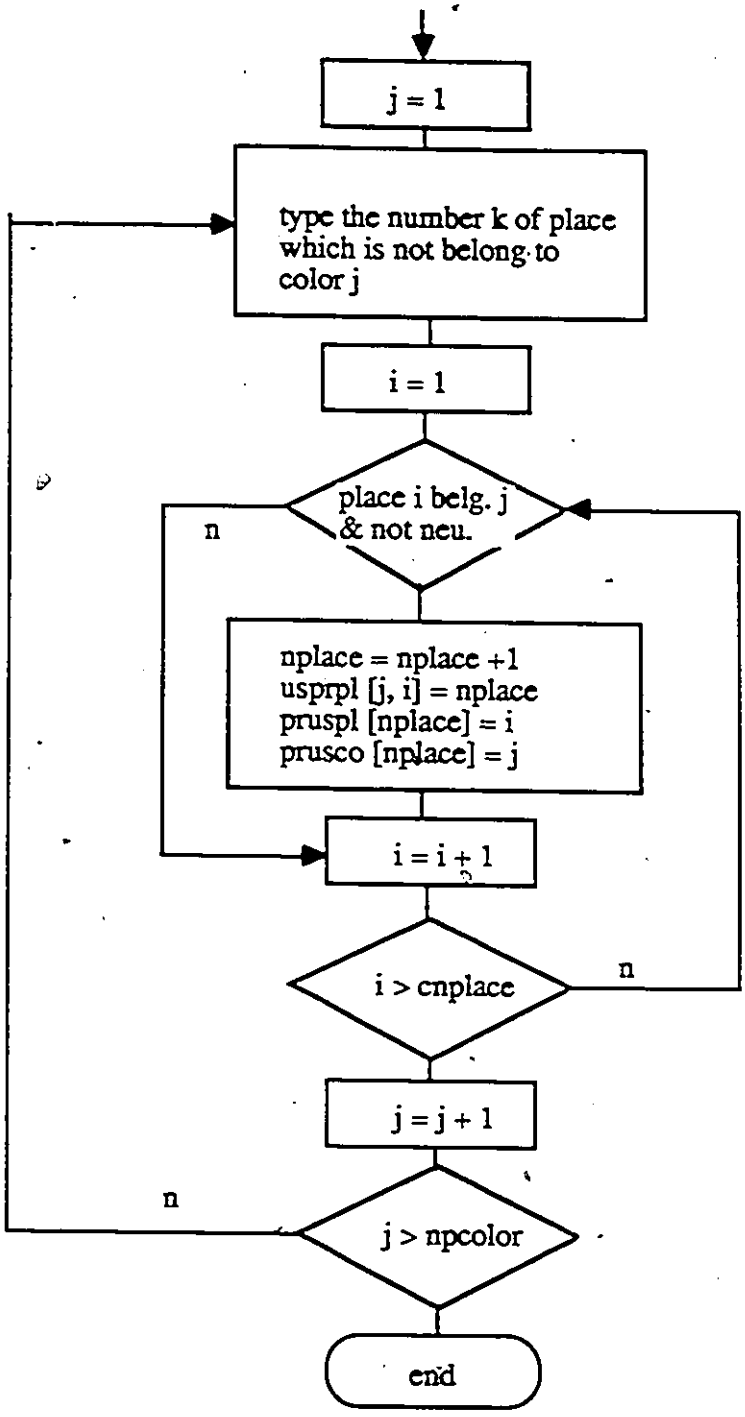


Procedure placetransform

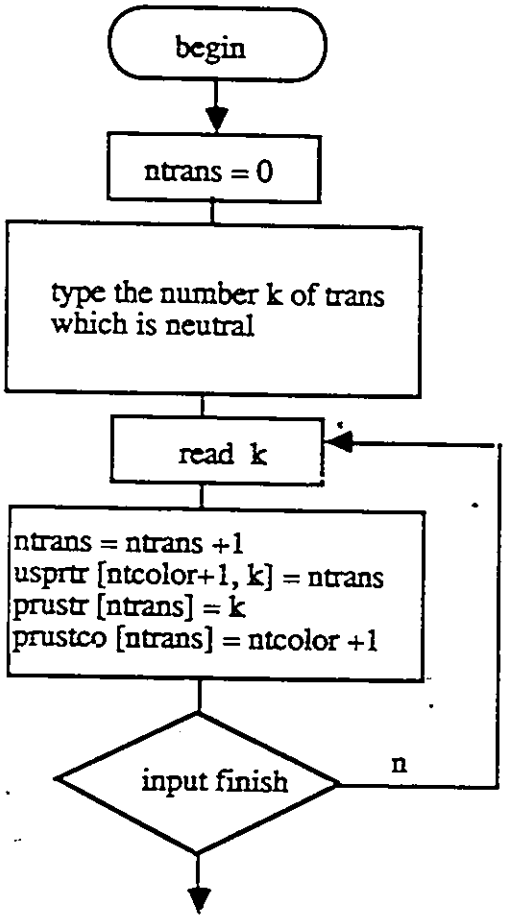


continued

Procedure placetransform (continued)

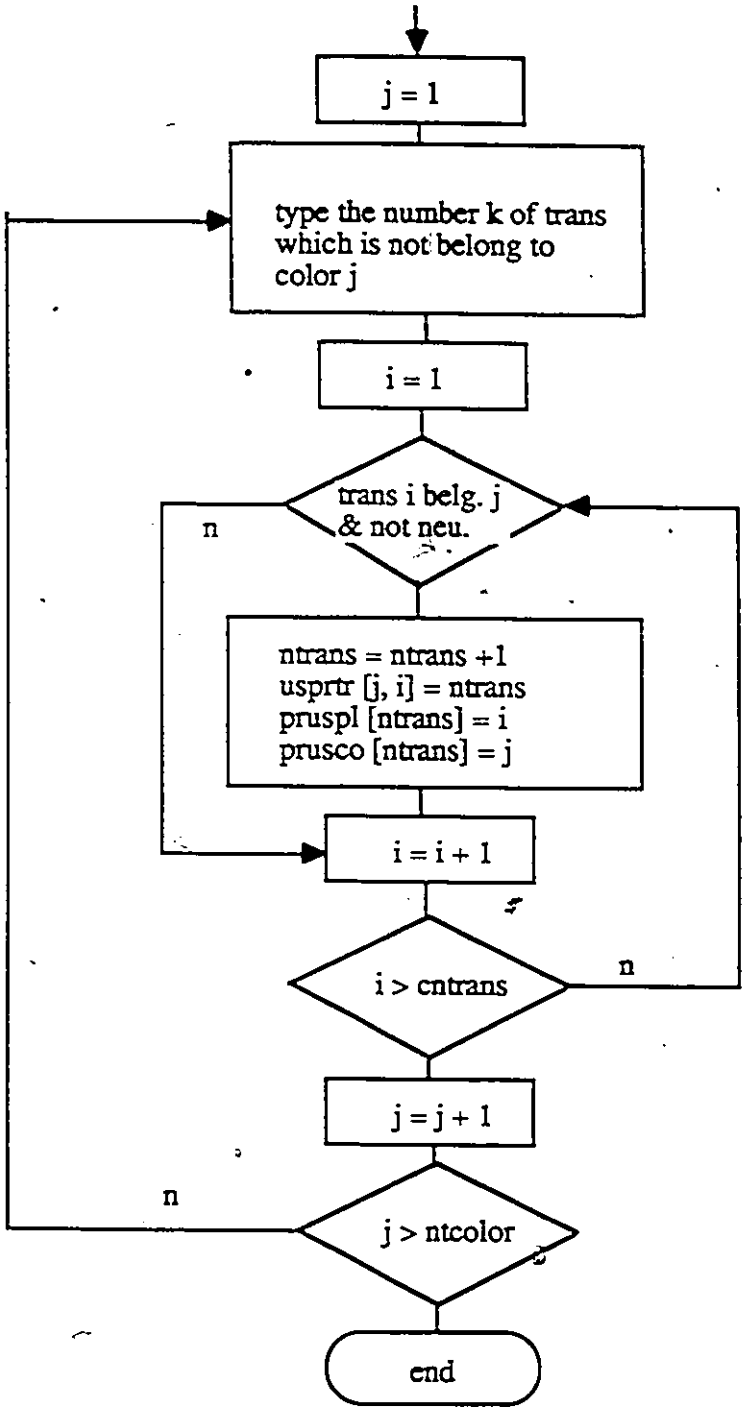


Procedure transtransform

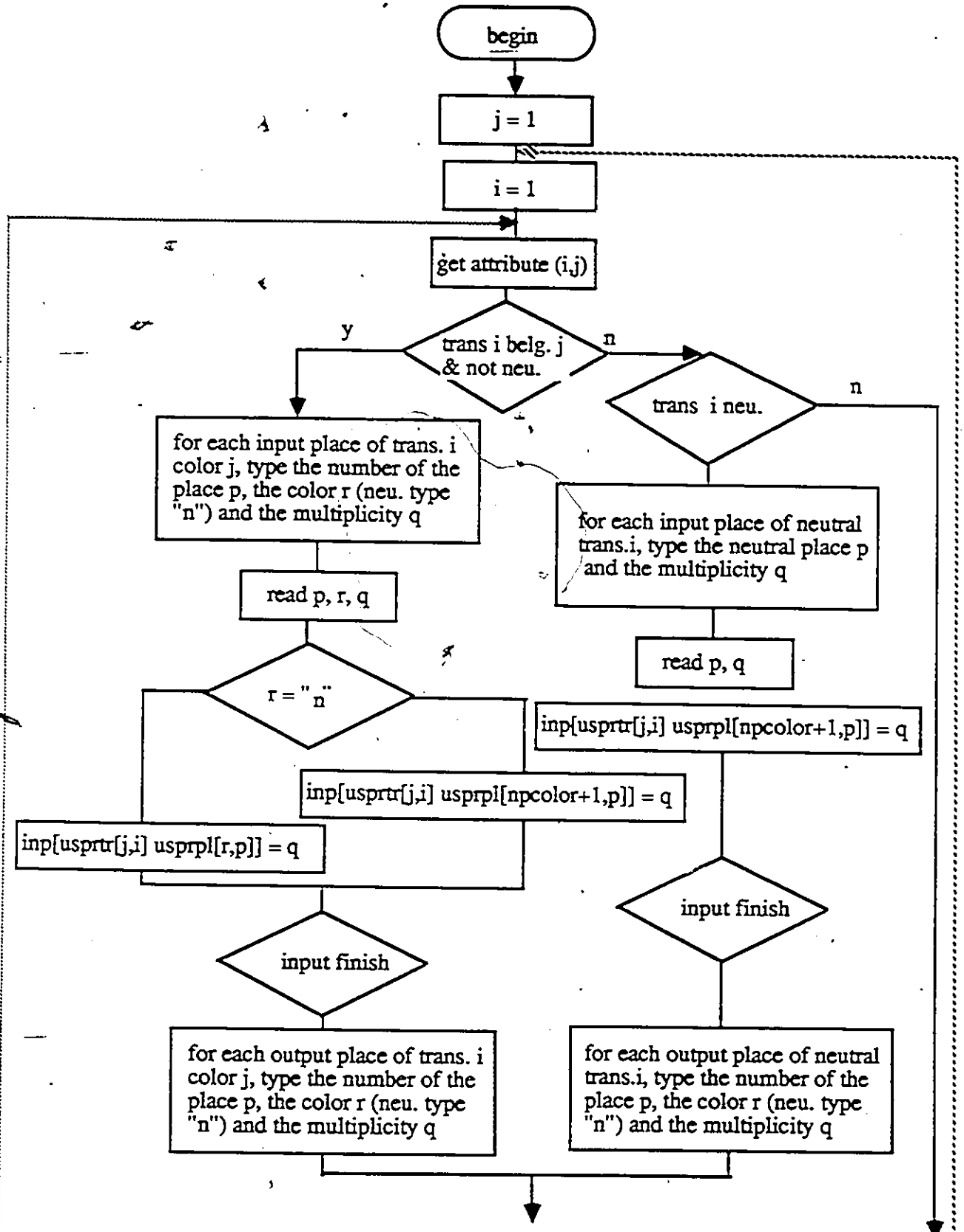


continued

Procedure transtransform (continued)

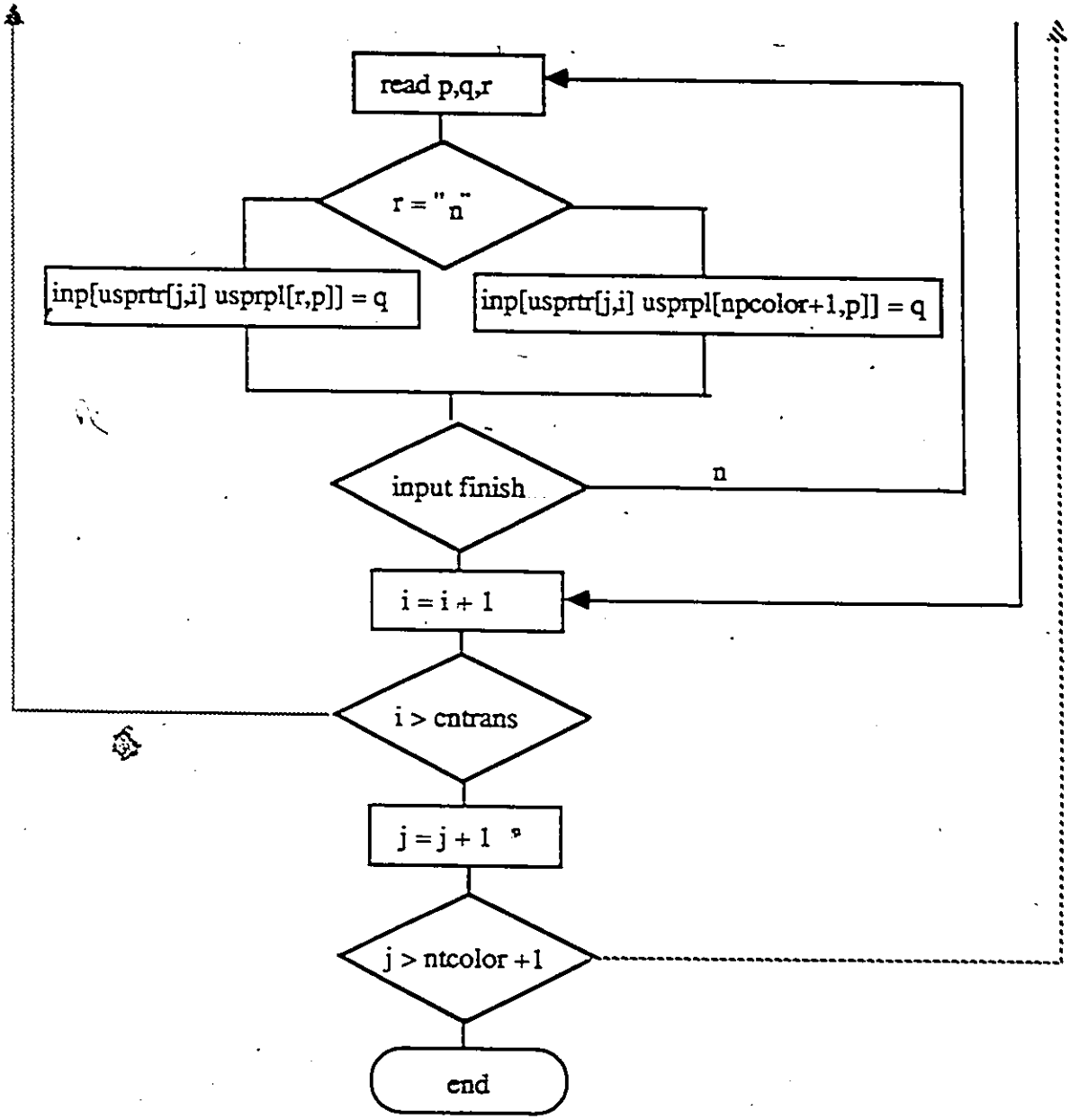


Procedure getinoutput

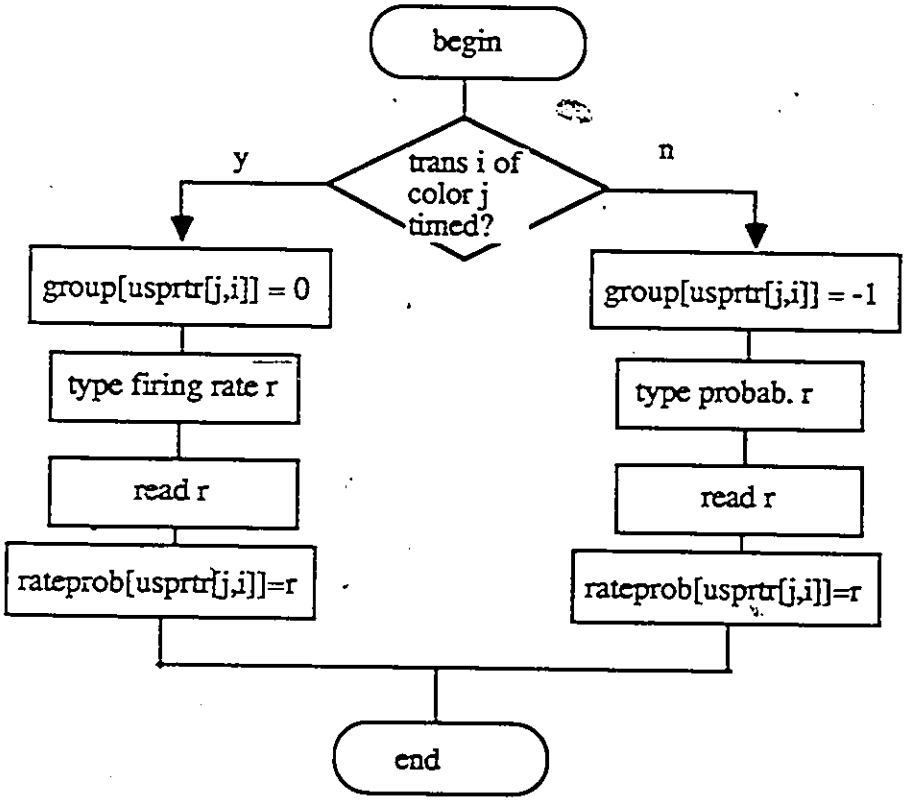


continued

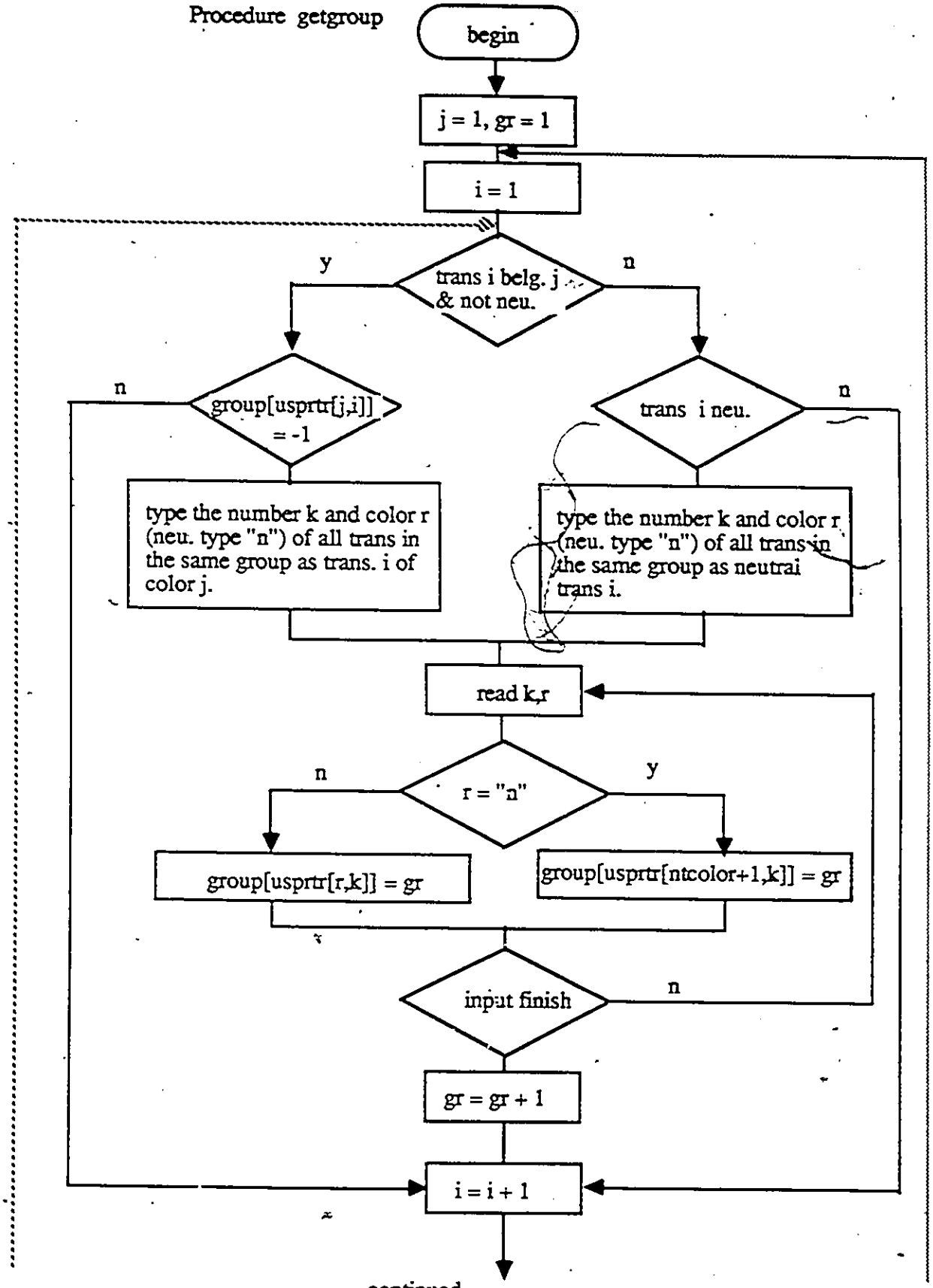
Procedure getinoutput (continued)



Procedure get attribute (i,j)

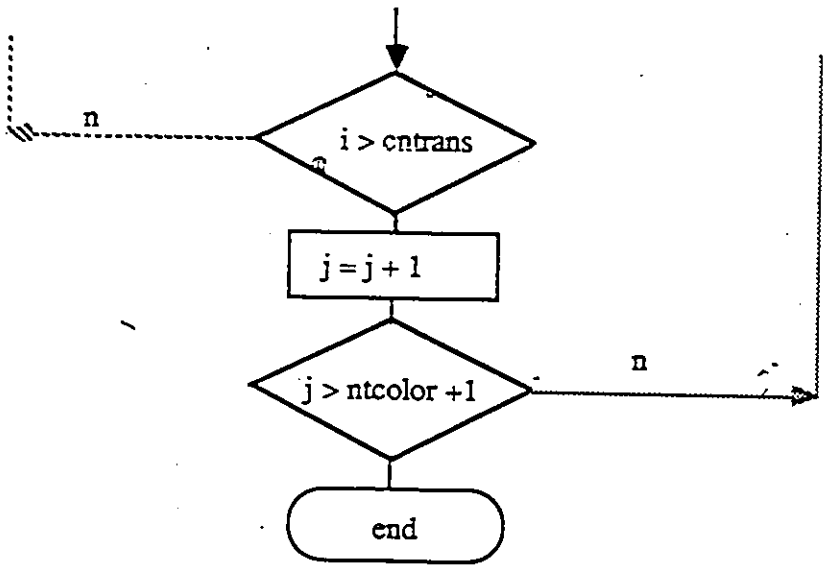


Procedure getgroup

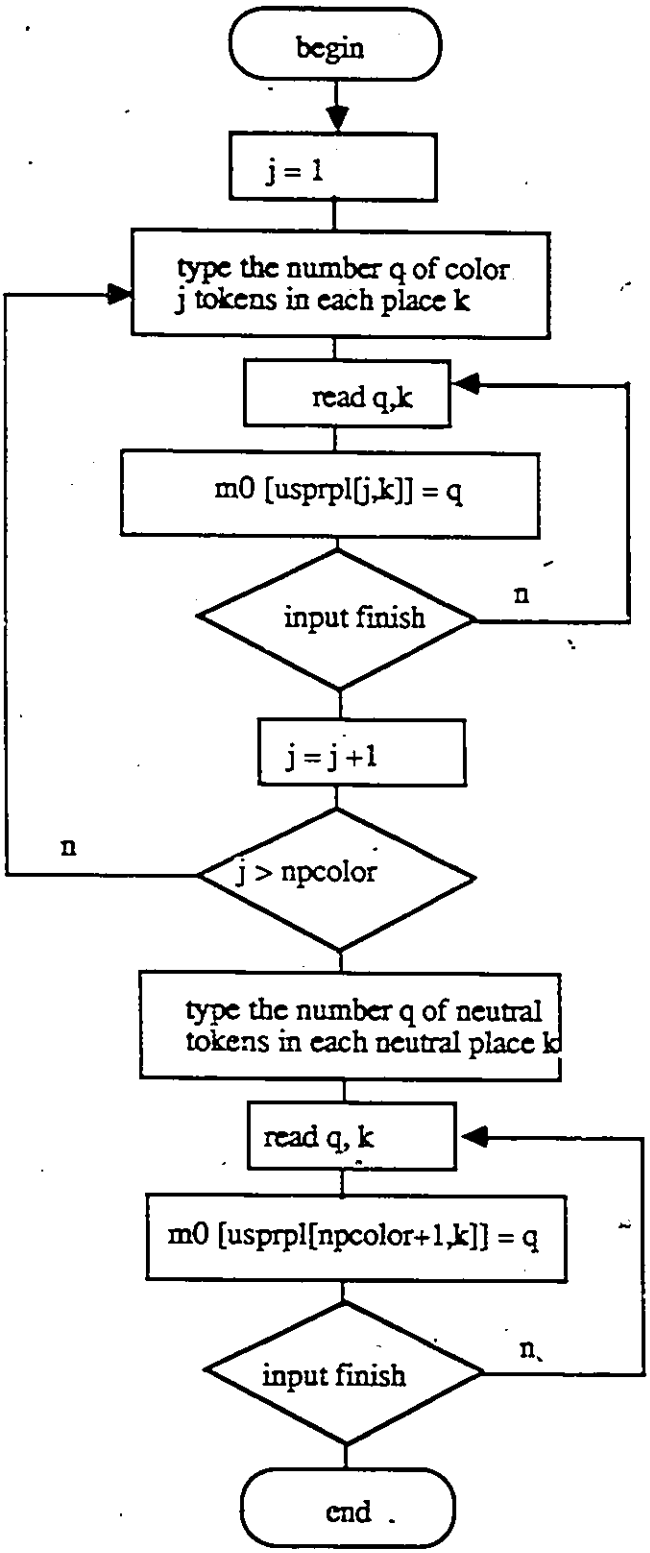


continued

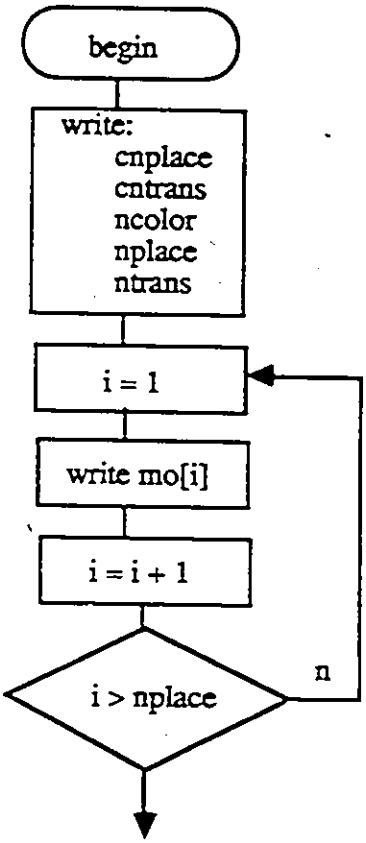
— Procedure getgroup (continued)



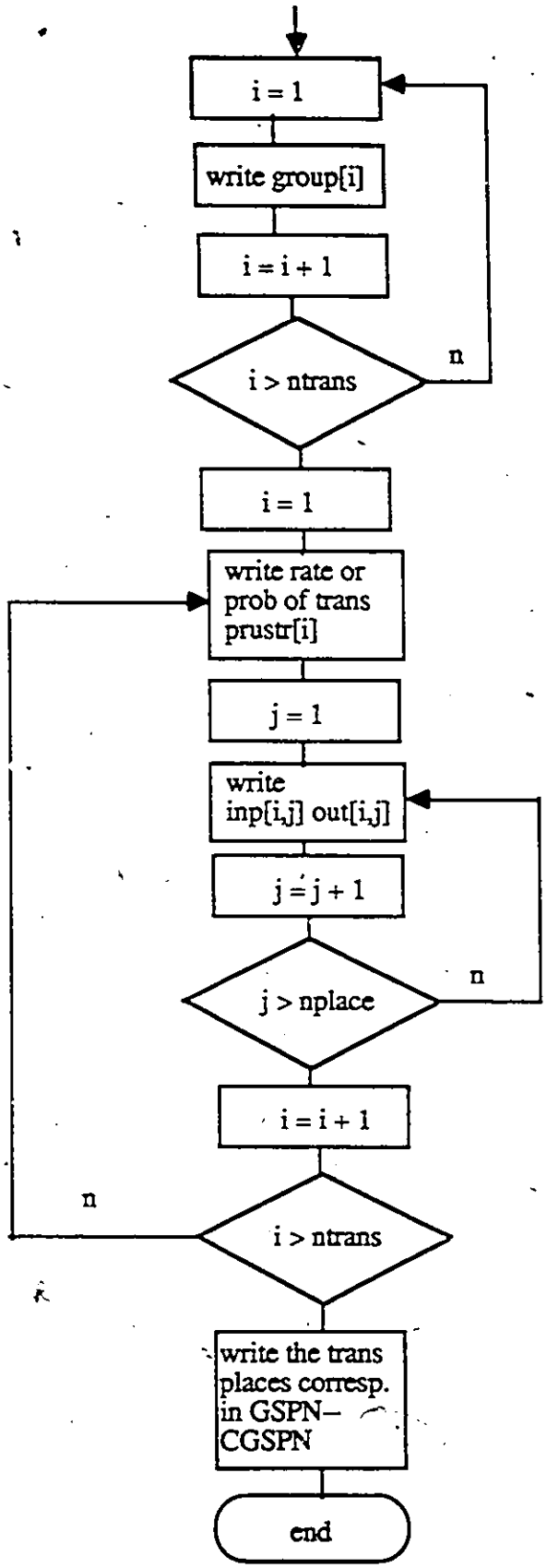
Procedure getinitial



Procedure writefile



continued


i = 1
write group[i]
i = i + 1
i > ntrans
n
i = 1
write rate or prob of trans prustr[i]
j = 1
write inp[i,j] out[i,j]
j = j + 1
j > nplace
n
i = i + 1
i > ntrans
n
write the trans places corresp. in GSPN– CGSPN
end

REFERENCES

[AJMO 84] M. A. Marson " A Class of Generalized Stochastic Petri Nets for the Performance Evaluation of Multiprocessor Systems" ACM Transactions on Computer System. May, 1984.

[AJMO 85] M. A. Marson " A Software Tool for the Automatic Analysis of Generalized Stochastic Petri Net Models" International Conference on Modelling Techniques and Tools for Performance Analysis. 1985.

[AJMO 86] M. A. Marson,et al, "Performance Models of Multiprocessor Systems". The MIT Press. 1986.

[BCMP 75] F. Baskett, K. M. Chandy, R. R. Muntz, F. G. Palacios "Open, Closed, and Mixed Networks of Queues with Different Classes of Customers". Journal of the ACM, Vol. 22 No. 2, April 1975.

[BOUD 85] J. Y. Le Boudec " A BCMP Extension to Multiserver Stations with Concurrent Classes of Customers". INRIA Rapports de Recherche No. 390, Mar. 1985.

[BRAD 65] P. T. Brady " A Technique for Investigating on-off patterns of speech" Bell System Technical Journal. 44. Jan. 1965.

[BRUE 80] S. C. Bruell, et al "Computational Algorithms for Closed

Queueing Networks" Operating and Programming System Series, Elsevier North - Holland, New York, 1980.

[BRUE 85] S. C. Bruell "Throughput Bounds for Generalized Stochastic Petri Nets" International Workshop on Timed Petri Nets. 1986.

[BRUN 86] H. Bruneel, M. Moeneclaey "On the Throughput Perfromance of Some Continuous ARQ Strategies with Repeated Transmissions" IEEE Transactions on Communications Vol. Com. 34 No. 3. March 1986.

[CONW 85] A. E. Conway, N. D. Georganas "RECAL – A New Efficient Algorithm for the Exact-Analysis of Multiple Chain Closed Queueing Networks" ACM SIGMETRICS Conference on Measurement and Modelling of Computer Systems Aug. 1985.

[DIAZ 82] M. Diaz " Modelling and Analysis of Communication and Cooperation Protocols Using Petri Net Based Models" Computer Networks, June 1982.

[ENGE 86] T. R. Engelbrecht " Predicting Protocol Performance From a Meta-Implementation" Protocol Specification, Testing, and Verification.V. IFIP 1986.

[GRUB 83] J. G. Gruber and N. H. Le. "Performance Requirements for Integrated Voice/Data Networks", IEEE Journal On Selected Areas in Communications. Vol. SAC -1, No. 6, Dec. 1983.

[HOLL 85] M. A. Holliday, et al " A Generalized Timed Petri Net Model for Performance Analysis" International Workshop on Timed Petri Nets. 1986.

[ISAA 76] D. L. Isaacson and R. W. Madsen " Markov Chains, Theory and Applications", Wily 1976.

[KEKR 77] H. B. Kekre and C. L. Saxena "Three State Markov Model of Speech on Telephone Lines and Optimal Utilization of Communication Systems by TASI Technique" Computer and Electrical Engineering Vol.4. 1977.

[KLEI 75] L. Kleinrock "Queueing Systems" Vol. 1 Theory. John Wiley & Sons, Inc. 1975.

[KRIT 85] P. S. Kritzinger " Analyzing the Time Efficiency of a Communication Protocol" Protocol Specification, Testing, and Verification.IV. IFIP 1985.

[KRIT 86] P.S. Kritzinger " A Performance model of the OSI Communication Architecture" IEEE Transactions on Communications Vol. C- 34 No. 6. June 1986.

[LAVE 83] S.S. Laverberg "Computer Performance Modiling Handbook" New York: Academic, 1983.

[LAZA 87] A. A. Lazar, et al. " Markovian Petri Net Protocols with Product Form Solution". Proceedings IEEE INFOCOM'87.

[LITT 61] J. D. C. Little " A Proof of the Queueing Formular $L=\lambda W$"

Operations Research. 9. 1961.

[MERL 76] J. A. Merlin " Recoverability of Communication Protocols – Implications of a Theoretical Study" IEEE Tranctions on Communications Vol. Com. 24 No. 9. Sept. 1976.

[MITC 86] L. C. Mitchell "End to End Performance Modelling of Local Area Networks" IEEE Journal on Selected Areas in Communications Vol. SAC - 4 No. 6 Sept. 1986.

[MOLL 81] M. K. Molloy "On the Integration of Delay and Throughput Measures in Distributed Processing Models" Ph.D dissertation, Univ. of California, Los Angeles 1982.

[MOLL 82] M. K. Molloy " Performance Analysis Using Stochastic Petri Nets" IEEE Transactions on Computers Vol. C-31 No. 9 Sept. 1982.

[MOLL 85] M. K. Molloy " Fast Bounds for Stochastic Petri Nets" International Workshop on Timed Petri Nets. 1986.

[MURA 87] M. Murata "Two-layer Modeling for Local Area Networks" Proceedings IEEE INFOCOM'87.

[NOE 73] J. D. Noe, et al " Macro E-nets representation of parallel systems" IEEE Transactions on Computers Vol. C-22 No. 8 Aug. 1973.

[PAPO 84] A. Papoulis "Probability, Random Variables, and Stochastic Processes" McGraw-Hill, Inc. 1984.

[PETE 81] J. Peterson "Petri Net Theory and the Modeling of Systems". Prentice Hall, Inc. 1981.

[PETR 66] C. A. Petri, "Communication with Automata", Ph.D thesis, Technical Report RADC-TR New York, Jan. 1966.

[REIS 79] M.Reiser "A queueing network analysis of Computer Communication Networks with Window Flow Control" IEEE Transactions on Computers Vol. C-27 Aug. 1979.

[REIS 80] M. Reiser, et al " Mean Value Analysis of Closed Multichain Queueing Networks" Journal of the ACM, Vol.27, April 1980.

[REIS 86] M. Reiser " Communication System Models Embedded in the OSI-Reference Model, a Survey" Computer Network and Performance Evaluation, IFIP, 1986.

[RUDI 83] H. Rudin " From Formal Protocol Specification Towards Automated Performance Prediction" 3rd International Workshop on Protocol Specification, Testing and Verification, Ruschlukon, Switzerland, May 1983.

[RUDI 85] H. Rudin " An improved Algorithm for Estimating Protocol Performance" Protocol Specification, Testing, and Verification. IV. IFIP 1985.

[SAUE 81] C. H. Saucer and K. M. Chandy " Computer Systems Performance Modelling" Prentice-Hall Inc. 1981.

[TEO 85] E. Teo " Report of an Integrated Data Voice Station". Computer Communications Laboratory, University of Ottawa, 1985.

[TEO 86] E. Teo, P. Mui, N. D. Georganas " Lan Protocols for Digital Voice Data Integration" Computer Communications. vol. 9 No. 6 Dec. 1986.

[TRIV 82] K. Trivedi "Probability and Statistics with Reliability, Queueing, and Computer Science Applications" Prentice -- Hall, 1982.

[WHEE 85] G. R. Wheeler, et al "Protocol Analysis Using Numerical Petri Nets" Lecture Notes in Comptuter Science Vol. 222, 1985.

[ZENI 85a] A. Zenie " Colored Stochastic Petri Nets" International Workshop on Timed Petri Nets. Torino, Italy, July 1985.

[ZENI 85b] A. Zenie "Les Reseaux de Petri Stochastiques Colores" Rapport MASI Issue No. 59, Universite P. M. Curie, Paris, April/ May 1985.

[ZUBE 80] W. M. Zuberek " Timed Petri Nets and Preliminary Performance Evaluation" Proc. IEEE 7th Ann. Symp. Compt. Arch. 1980.

ABSTRACT

Modelling and evaluation of protocol performance for integrated systems are of very high current interest. Most of these performance studies consider models of only one specific protocol layer. Layers above the one considered are usually modeled as workload sources, while those below are considered as transmission models for analytical tractability. Resource contention from different layers, cooperation, synchronization and concurrency between different protocol layers, as well as the different protocol needs of digitized voice and a variety of data traffics, which constitute an important part of an integrated system protocol specification, are not considered.

In this thesis, we propose the use of Colored Generalized Stochastic Petri Nets (CGSPN) as a tool for performance modelling of integrated networks. We also extend the CGSPN by introducing the new concepts of "priority firings" and "neutral tokens". We show that the resulting extended CGSPN is still a Markov process. In order to obtain numerical results from the CGSPN, we build a PASCAL programme extending the software package GSPNA, which is used for analyzing Generalized Stochastic Petri Nets. We believe that all these extensions are major contributions to both the Petri Net theory and its applications.

As an application, we show how to use the extended CGSPN to model an integrated data voice system and how to obtain the numerical results. A CGSPN model is built, with different color tokens representing data and voice and neutral tokens representing CPU resources respectively. Priority

levels are assigned to transitions representing the different priorities of data and voice packets. The performance of the integrated system protocol is evaluated by numerically solving the CGSPN model.