DESIGNING OF A SIMULATION TOOL TO ANALYSE THE DINING PHILOSOPHER'S PROBLEM BY PETRI NET ANALYSIS

A thesis submitted in partial fulfillment of the requirements for the degree of

Bachelor of Technology

in

Computer Science and Engineering

by

Bineeta Kachhap

(Roll no. 107CS017)

Under the guidance of :

Prof. S.K.Rath

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, Orissa, India

National Institute of Technology Rourkela

Certificate

This is to certify that the project entitled, '**DESIGNING OF A SIMULATION TOOL TO ANALYSE THE DINING PHILOSOPHER'S PROBLEM BY PETRI NET ANALYSIS**' submitted by **Bineeta Kachhap** is an authentic work carried out by them under my supervision and guidance for the partial fulfillment of the requirements for the award of **Bachelor of Technology Degree** in **Computer Science and Engineering** at **National Institute of Technology, Rourkela**.

To the best of my knowledge, the matter embodied in the project has not been submitted to any other University / Institute for the award of any Degree or Diploma.

Date - 12/05/2011

Rourkela

(Prof. S.K.Rath)

Department of Computer Science and Engineering

Acknowledgments

I express my profound gratitude and indebtedness to **Prof. S.K.Rath**, Department of Computer Science and Engineering, NIT, Rourkela for introducing the present topic and for his inspiring intellectual guidance, constructive criticism and valuable suggestion throughout the project work.

Finally, I would like to thank my parents for their support and motivation to complete this project.

Date - 12/05/2011
Rourkela

Bineeta Kachhap

Abstract

Concurrent Systems are those where time is shared by each smaller module. Concurrency is the major problem in many of the real world system. It generally arises due to synchronization problem or improper scheduling between the modules. Such kind of sophisticated and large system is modelled for their analysis and study. Petri Nets is a graphical and mathematical tool which helps in modelling these systems that are concurrent, distributed, parallel, nondeterministic, asynchronous and stochastic.

The Dining Philosophers Problem is a classic Synchronization problem of the multiprocessor systems. The project primarily includes the simulation of an example of multiprocessor system and designing and creation of a special simulation tool for Dining philosophers Problem by Petri Net. The tool is created in JAVA. The tool analyses the graph dynamically drawn consisting of places which represents the philosopher can think or eat and availability of the chopsticks and the transition represents whether a philosopher can pick up or return the stick. The prime focus is given to the User-Define Time constraint for a particular philosopher. The simulation tool guides the user at each and every step and the solutions given by the tool ensure a freedom of deadlock and in process coordination.

Purpose of the project

The Dining Philosophers Problem is a classic Synchronization problem of the software system. In this system five philosophers are dining; they are provided with five chopsticks, they can either think or eat. The problem symbolises working of resource management in a system. The chopstick stand for the resource and the philosopher stands for the process. Availability of the chopstick denotes the availability of the resource for the particular process to execute. The synchronizing problem helps in preventing deadlock situation in dynamic resource allocation.

A Petri Net graph is formed that analyses the problem of synchronization and helps in process coordination. The corresponding Petri Net model consists of places which represents the philosopher can think or eat and availability of the chopsticks and the transition represents whether a philosopher can pick up or return the stick. Analysis the Dining Philosophers Petri Net model is done by the Incidence- Matrix and State-Equation analytic technique. It is algebraic technique where the Incidence- Matrix is formed by calculating the weight of the output arc and the input arc for a transition and State-Equation is formed the unit control vector.

The thesis contains the brief discussion about Petri Net and their models, basic characteristic of a Petri Net are being discussed. The project primarily includes the analysis of the Dining philosophers Problem by Petri Net, designing and creation of a SIMULATOR. The tool is created in JAVA by using graphics. Workings of the simulator are being shown by taking a simple example of multi-processor system. The places, transition, token numbers and weighted arcs are added and subsequently the tool takes up the model as an input. The tool is built up to stores the value for each place, arc an forms the Incidence Matrix. On the completion of the model the tool asks the user for the transition to be fired and shows the next state accordingly calculated by the State-Equation formula. The prime focus of the tool is the User-Define Time constraint given to a particular philosopher in the Dining Philosophers Problem. As the philosophers picks up the stick, a timer is set up that describes at what time the philosopher would return the stick. The time is calculated by storing the time of the system. All through the process simulation, the tool guides the user at each and every step by generating a

message whether the transition is fireable, the time limit for returning the stick is completed and what are the next fireable states. Test cases are being taken up to show the synchronization of these problems.

The messages generated, helps the user to fire a valid transition so as to prevent deadlock and the solution given by the tool ensure dead lock free process coordination.

Contents

List of Figures

1 PETRI NET

1.1 INTRODUCTION

Petri Net was introduced by Carls Adam Petri in early 1961 during his PhD Dissertation [6]. Petri started his scientific career with his dissertation "Communication with Automata", which he submitted to the Science Faculty of Darmstadt Technical University, West Germany in July, 1961. He defended his thesis there in June, 1962[6].

1.2 OBJECTIVE

A large software system comprises of smaller modules that have different functionalities. Each smaller module executes a particular process for the overall execution. These Software systems can be classified into various categories depending upon time, speed or execution etc. For instance, basing on time we have sequential system and concurrent systems. Concurrent Systems are those which share time and that affect the performance. Concurrency is the major problem in many of the real world system. It generally arises due to synchronization problem or improper scheduling between the modules. Such kind of sophisticated and large system is modelled for their analysis and study. Petri Nets helps in modelling these systems that are concurrent, distributed, parallel, nondeterministic, asynchronous and stochastic.

Petri Net is a graphical and mathematical tool use to solve complexity of various kinds of systems. The Petri Net Model of a system comprises of places, transition, tokens and weighted arc and it provides a simple solution about the synchronization and process coordination.

1.3 APPLICATION

[7]Be it any field like manufacturing systems, communication network, software design, workflow management, data analysis, concurrent programming, reliability engineering, diagnosis, controlling discrete event systems, KPN modelling, Petri Net helps in modelling, simulation, control and performance analysis.

1.4 TERMS AND DEFINITIONS

FORMAL DEFINITION: [1]A petri net is a five-tuple, PN = [P, T, F, W, M0] where:

P = [p1, p2, ..., pm] is a finite set of place and cardinality of P = m.

T = [t1, t2, ..., tn] is a finite set of transition and cardinality of T = n.

F = (P X T)(T X P) is the set of arcs(Flow relation).

W: F → [1, 2, 3, 4, ...] is the weight function.

M0: P→ [0, 1, 2, 3, ...]

P union T is an empty set and P intersection T is not an empty set

[1]A Petri Net structure N = (P, T, F, W) without any specific initial marking is denoted by N.

A Petri Net with the given initial Marking is denoted by (N, M0). [1]The behaviour of

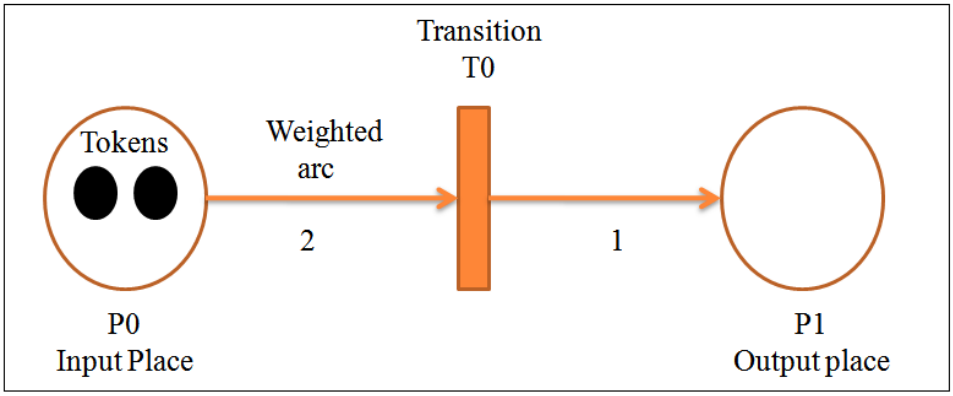

Figure 1: Petri Net Model

the system is determined by the way their initial state is and the change in the state. In order to simulate the dynamic behaviour of the system a state or marking in a petri net is changed according to the TRANSITION RULE; the rule states:

a) A transition T is said to be enabled if each input place P of T is marked with at least W(P, T) tokens where W(P, T) is the weight of the arc from P to T.

b) An enabled transition may or may not fire, depending on whether or not the event actually takes place.

c) A firing of an enabled transition removes W(P, T) tokens from each input place P of T and adds W(T, P) tokens to each of the output place P of T where W(T, P) is the

weight of the arc from T to P.

Example: In the Figure1, P0 is the INPUT PLACE and P1 is the OUTPUT PLACE of the transition T0. The WEIGHT of the input arc is 2 and the output arc is 1 and there are 2 TOKENS present in input place. By transition rule T0 is fireable as W (P0, T0) = 2 and there are two token present in the input place. The initial state of the above petri net is M0= [2, 0]. Firing of T0 gives the state M1=[0,1] as the output arc places 1 token to P1 as shown in Figure2.

TERMS

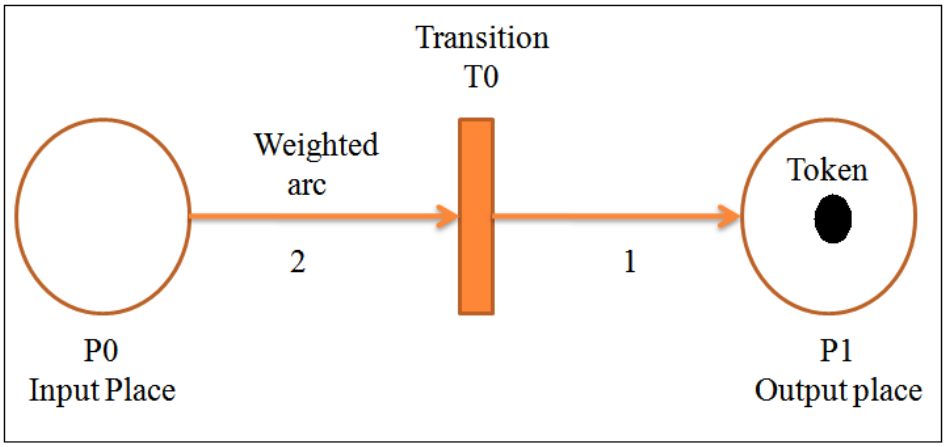

Figure 2: State after firing of T0

[1]A transition without any input place is called a SOURCE TRANSITION and one without an output place is the SINK TRANSITION. A source transition is unconditionally enabled and a sink transition consumes the token but does not produce any. A pair of of place P and a transition T os caaled a SELF-LOOP if P is both the input and the output place of the transition T. A petri Net is called as PURE if there are no self-loop. A petri net is called ORDINARY if all the arc weights 1. A petrinet is refered as INFINITE CAPACIY NET if the places can accommodate unlimited numbers of tokens and [1] for modelling many kinds of system, an upper limit of number of token is set for the places that it can hold. This kind of the petrinet are refered as FINITE CAPACITY NET. In

finite capacity net each places are associated with a capacity K(p),the maximum number of token place p can hold and there is an addition rul in the transition rule that the transition is not fireable if the tokens in outplace place exceed the capacity K(p). this capacity constraint rule is called as STRICT TRANSITION RULE[1].

2 PETRI NET MODELS

2.1 FINITE STATE MACHINE

[1]The finite machine diagram or the state diagram represents one of the subclasses of petri net model that is called as STATE MACHINE. A simple example of vending machine is given which takes a amount of coin for candy bar shown in the Figure 3. Each transition has exactly one input arc and one output arc. There are conflicts, decision or choices found in this state machine but not synchrnization of parallel activities.

2.2 PARALLEL ACTIVITES

Parallel or concurrent activites can be shown in the model. [1]Two transition are called to be parallel if they are causally independent i.e. firing of one transition does not affect another. These types of graph helps in showing concurrency but nor decision or conflicts. Example Figure 4.

2.3 DATAFLOW COMPUTION

Petrinet diagrams can show the flow of data. In the given example, Figure5, the data flow computation is being described where token in places represents the operand and transition represent the instructions.

2.4 COMMUNICATION PROTOCOL

A simple example of communication protocol between two processor is described in Figure 6. These petri net model often use behavioural properties like liveliness and safeness for the correction.

2.5 SYCHRONIZATION CONTROL

[1]In a multiprocesor or distributed processing-system, resources and information are shaed by several processor. The sharing is controlled and synchronized to have a cor-

rect operation.Petri net uses variety of synchronization methods like mutual-exclusion, readers-writer, producer-consumer problem. A simple example of readerwriter is given in the Figure 7 .

2.6 PRODUCER-CONSUMER SYSTEM WITH PRIORITY

A inhibitor arc is connects froma place to a transition is reprsented by the dashed lines terminating with a small circle as shown in the Figure 8. The corresponding figure shows that consumer A has priority over consumer B . The inhibitor arc disables transition when the input place has tokens and enables transition when the it has no tokens.these subclasses are called as EXTENDED PETRI NETS.

2.7 FORMAL LANGUAGE

When a sequence of transition firing generates strings of symbol,these set of strings generated by all possible firing sequence defines a formal language called the PETRI NET LANGUAGE. Refer to Figure 9.

2.8 MULTIPROCESSOR SYSTEM

A two bus multiprocessor is being described in Figure10 where the five places represents active processor, free buses, processors waiting for request to be accepted, processor having access to common memories and processor queued for common memory.

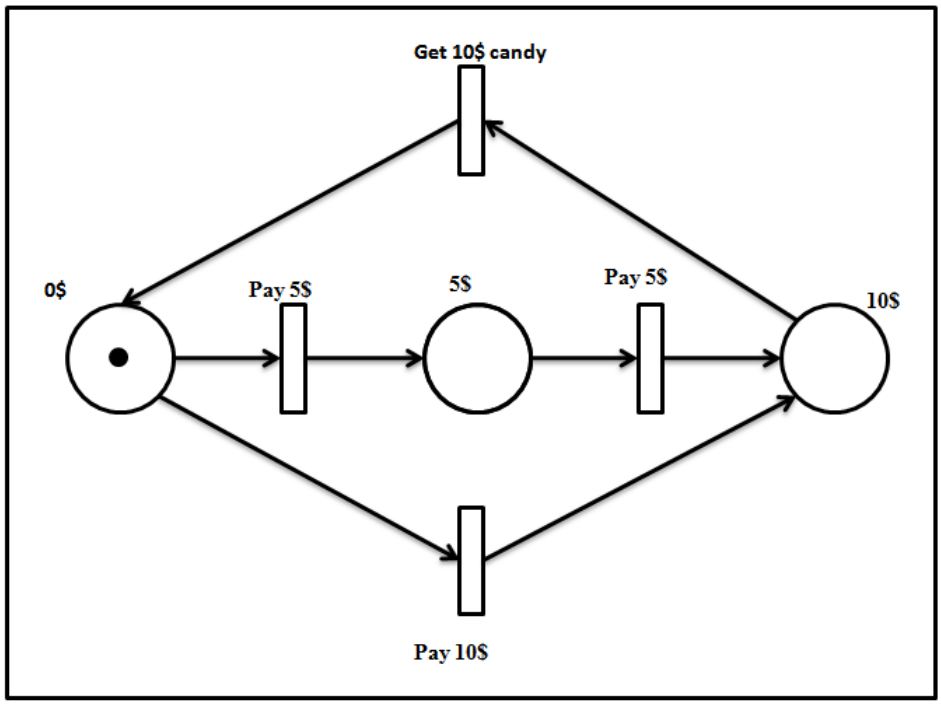

Figure 3: A Petri Net, State Machine representing a vending machine

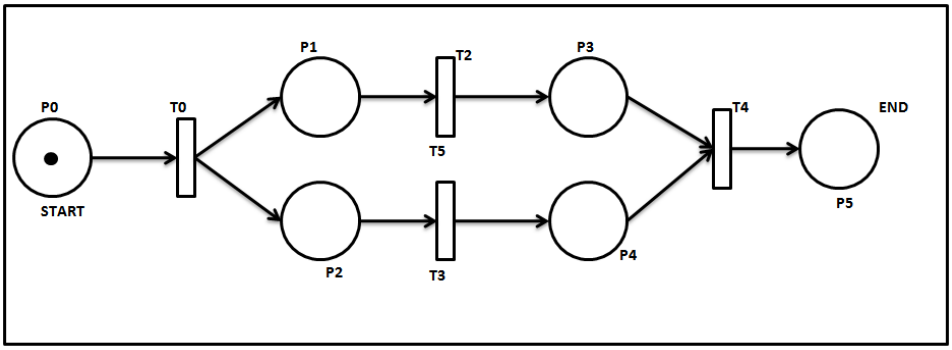

Figure 4: A Petri Net showing concurrency

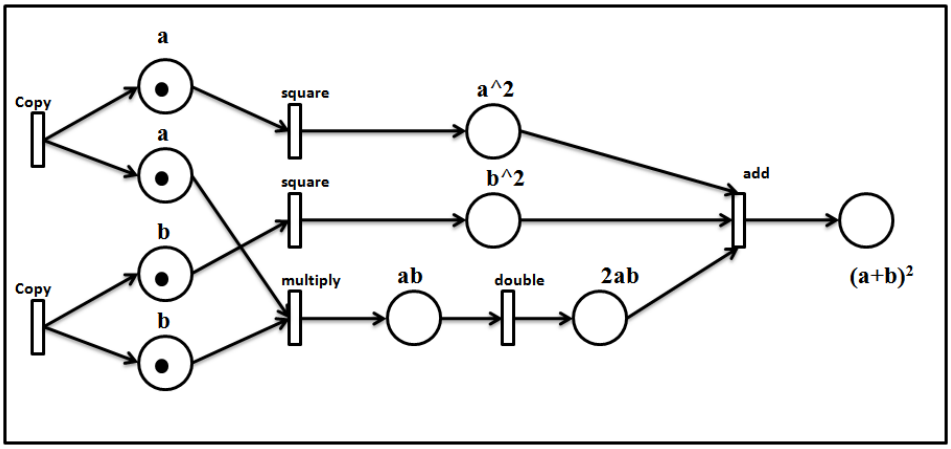

Figure 5: A Petri Net showing dataflow computation of $(a + b)^2$

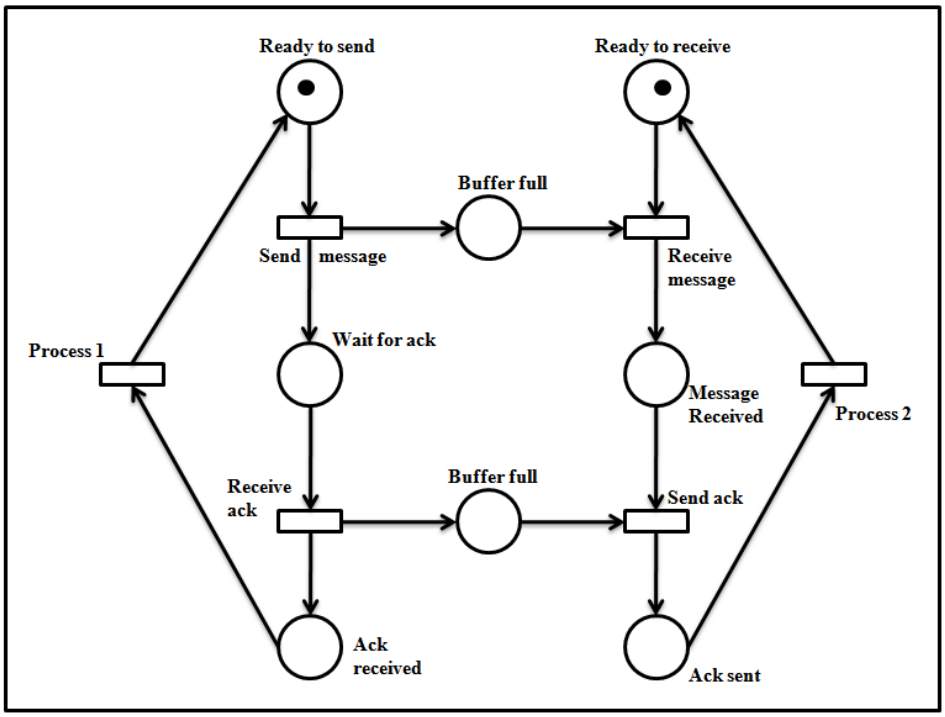

Figure 6: A Petri Net showing communication between two processor

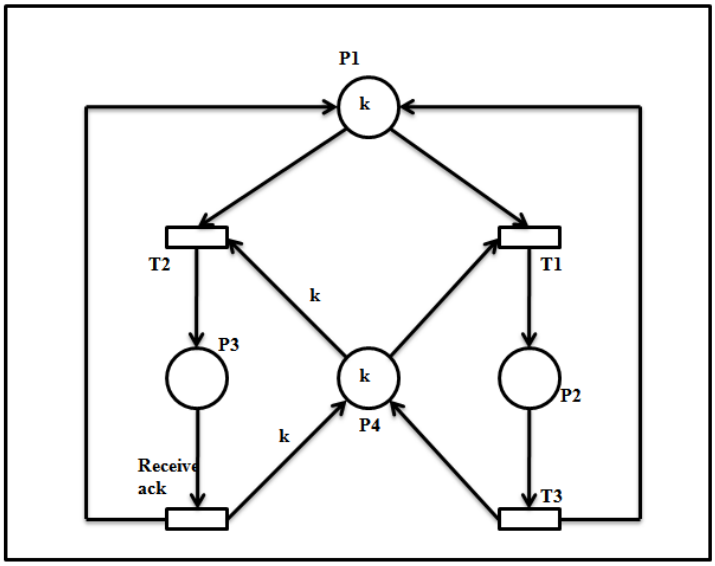

Figure 7: A Petri Net showing readers- writers system

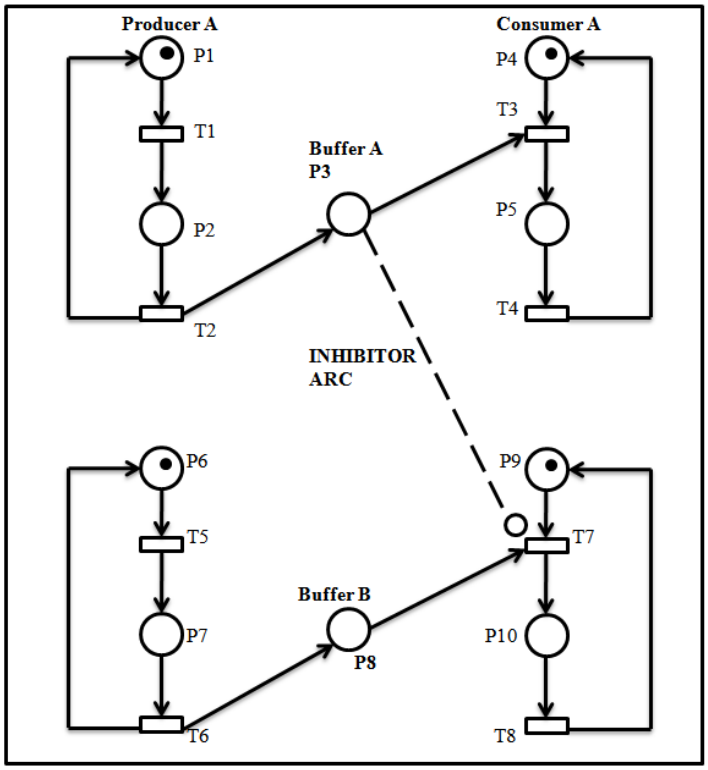

Figure 8: An Extended Petri Net with producers-consumer priority

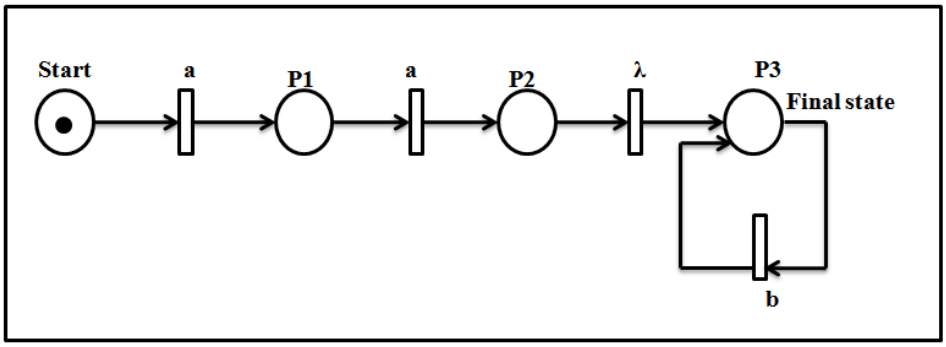

Figure 9: A Petri Net generating a language L= aab^n

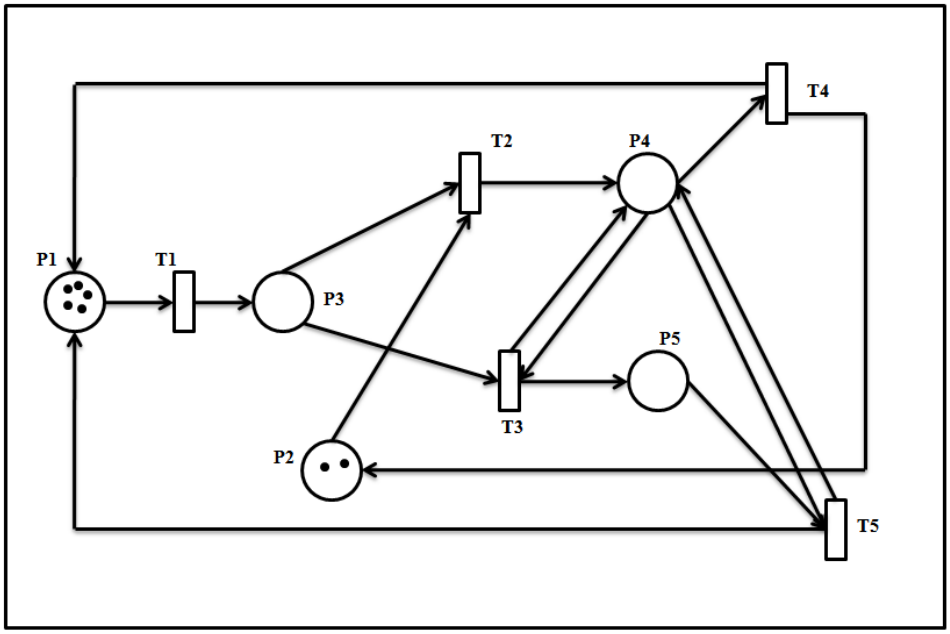

Figure 10: A Petri Net showing a multi processor system

3 SUBCLASSES

[1] Petri nets can be ordinary or non-ordinary,both have the same modelling power but are distinguised by their convenience nad modelling efficiency.Certain pre-sets and post-sets are used like $\bullet$T = set of all input place to the transition T, T$\bullet$ = set of all output place to transition T, $\bullet$P = set of all input transition of place P , P$\bullet$ = set of all output transitions to place P.

Sub-classes of Petri-net are 3.1 State-Machine: As discussed earlier, an ordinary petri ne twhere each transition has one number of input place and output place, $|\bullet T| = |T\bullet| = 1$.

3.2 Marked-Graph: A ordinary sub-class of petri net where the number of input transition and output transition of a place is 1. $|\bullet P| = |P\bullet| = 1$.

3.3 Free-choice net: A ordinary petri net where every incoming and outgoing arc is unique i.e. for all P1, P2 belonging to P, P1$\bullet$ intersection P2$\bullet \neq \phi \Rightarrow |P1\bullet| = |P2\bullet| = 1$.

3.4 Extended free-choice net: A ordinary Petri Net where , for all P1, P2 belonging to P, P1$\bullet$ intersection P2$\bullet \neq \phi \Rightarrow |P1\bullet| = |P2\bullet|$.

3.5 Symmetric choice: A ordinary Petri Net where for all P1, P2 belonging to P, P1$\bullet$ intersection P2$\bullet \neq \phi \Rightarrow$ P1$\bullet$ is a subset of P2$\bullet$ or P2$\bullet$ is subset P1$\bullet$.

4 BEHAVIOURAL PROPERTIES

- Liveliness: A petri net is live is it does not consists of any deadlock situation. A transition T in Petri net is said to be L0-live(Dead): if T cannot be fire in the sequence. L1-live(Potentially fireable): if T can be fired at least once. L2-live: if T can be fired for a given number of times. L3-live: if T appears infinitly in the firinf sequence. L4-live:if T is L1-live for every marking in R(M0).

- Reachability: Given a Petri Net PN with initial marking as M0, then we say Mn is reachable from M0 iff after a sequence of firing M0 results to Mn. Represented as R(M0) = Mn.

- Boundedness: Given a Petri Net PN with initial marking M0, PN is said to be k-bounded if the number of token doesnot exceed the capacity k.

- Coverability: A marking M1 is said to cover M2, iff M1>=M2 and R(M0) = $(M1, M2)$

- Reversibility and home state: A Petri Net PN is said to be reversble if each of the marking M belongs to R(M0) and M0 belongs to R(M). In such cases a initial state is defined as the home state. Suppose M1 is the home state then each marking M belonging to R(M0), M1 belongs to R(M).

- Persistance: A petri net is said to be persistance if firing of one transition does not disable the other transitions.

- Synchronic distance: the distance between the two transitions are defined as
  here, sigma(T1) represents the total no of times T1 is fired.

$$d_{12} = \max | \sigma(T1) - \sigma(T2) |$$

- Faireness: [1]Two transition T1and T2 are said to be in bounded-fair if maximum number of times T1can fire and T2 is not firing. [3]And are said to be unconditionally fair if a every transition in sequence occurs infinitely.

5 THE ANALYTIC TECHNIQUE

5.1 INTRODUCTION

Modelling a system with Petri net provides a major strength for analysing the properties and problems associated with the concurrent systems.

The analysis method is classified into three groups:

1. Structure analysis

   - Siphon
   - Reduction and Decomposition technique.

2. State Space analysis

   - The Coverability tree method.

3. Algebraic technique

   - The Matrix-Equation approach.

5.2 PURPOSE OF ANALYSIS

Various Systems are model into Petri Net for their study and analysis. In some cases two different systems have the same Petri Net Model. Complexities of such Petri net model are difficult to study and implement. Different analytic techniques are implemented to study the complexity, to find out the feasible states or the reachability condition of the Petri net. The analytic techniques help in studying the behavioural properties of the Petri net such as liveliness, safeness, boundedness. Proper coordination of the modules only occurs if there is synchronization between them. Analysis methods help in implementation of Petri net model.

5.3 DIFFERENT ANALYTIC TECHNIQUES

5.3.1 STRUCTURAL ANALYSIS

A) REDUCTION AND DECOMPOSITION

Description: Analysing a large system by Petri Nets is difficult as a system may model into net comprising of several places and transition with numerous weighted arcs. These types of Petri Nets can be replaced and modified by simple reduction rules to model a simpler one. The transformed Petri Net formed is persistent and has all the properties of the original Petri Net. These rules help in analysing the properties namely analysing liveliness, safeness and boundedness in a simpler manner.

Method: The Transformation rules are:

i. Fusion of Series Places. (FSP)

ii. Fusion of Series Transition. (FST)

iii. Fusion of Parallel Places. (FPP)

iv. Fusion of Parallel Transition. (FPT)

v. Elimination of Self-Loop Places. (ESP)

vi. Elimination of Self-Loop Transition. (EST)

B) SHIPHON

[4] Given a Petri Net (PN) = (P, T, F, W, M0), then S is a subset of places and S is called a Trap (Siphon) iff S$\bullet$ is a subset of $\bullet$ S and $\bullet$ S is a subset of S$\bullet$. It means that firing of transition T deducts and adds same number token to the place P in S. Siphon plays an important role in Free-Choice Petri Net where $\bullet$ T1 intersection $\bullet$ T2 $\neq \phi$ => $\bullet$ T1 = $\bullet$ T2.[4] Literary means that if two transition have same input place then they share all the input places. A Free choice Petri Net is alive iff every non empty siphon contains an initially marked trap.

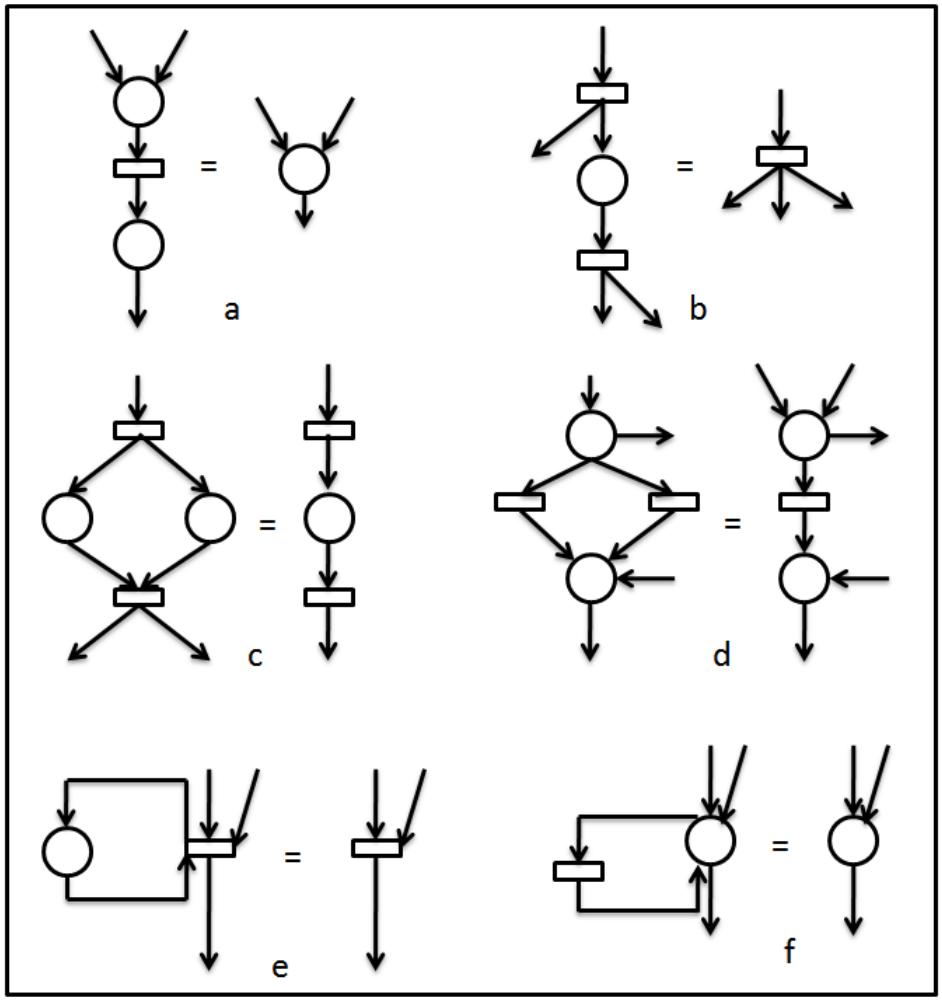

Figure 11: Diagrammatic representation of transformation rules

5.3.2 SPACE-STATE ANALYSIS

THE COVERABILITY TREE

Description: The method comprises of the formation of The Coverability Tree which involves essentially the enumeration of all reachable markings or their coverable marking from the initial marking. Basing on the coverability tree, different properties of the Petri Net graphs can be studied. It is simple and clear approach for analysing the Petri net. Given a Petri net (N, M0) with initial marking M0, a new marking is obtained by the no of enabled transition. This is an iterative method where a new marking reach outs to more number of markings. Each NODE represents the state of the Petri net and the ARC represents the enabled transition that is fired. The reachability graph has some limitation; this method can only be applied to bounded Petri Net models. If a system is not bounded then the graph grows infinitely. To solve this issue we make the tree finite by introducing a special symbol assumed to be infinity. For an integer n,
$w > n, w + n = w, w - n = w, w >= w$

Method: [1] The coverability tree is formed by the following algorithm:

1. Let the initial marking M0 be the root and label it as new.
2. If the new marking exist then

2.1 Select a new marking M.

2.2 If M is identical to a marking on the transition series path from the root till the M, then label it as old and then find a new marking.

2.3 If no transition are enabled at M the label it as dead-end.

2.4 If a transition is enabled at M, for each fireable transition do the following steps;

2.4.1 Obtain a temporary marking M1 that result from firing t at M .

2.4.2 In the transition series path from root to M if there exist another marking M2 such that M1 (p) >= M2 (p) for each place p and M1 not equal to M2 then M2 is coverable and replace M1 (p) by ω.

2.4.3 Create a node M1, draw an arc from M to M1 for transition t and label it as new.

A labelled directed graph G = (V, E) is formed from the coverability tree of the Petri Net (N, M0), where V is the set of all the nodes in the coverability tree and the arc set E is for transition fired.

Properties: [1] Some properties analysed by the coverability (reachability) tree;

1. If a Petri Net is bounded i.e. if the set of reachable states are finite then no nodes would contain the (infinity symbol).
2. If all nodes contain 0s and 1s then the Petri Net model is safe.
3. Absence of a transition in the coverability graph shows that the transition is dead.
4. If M0 reaches to M, then there exist a nodeM1 such that M>=M1.

Limitations: [4] The boundedness property is not completely analysed in this method. The infinite symbol in the coverability tree does not specify the total or the exact number of the reachable markings from the root node; this can be achieved only by the reachability graph which shows the number of the sets of reachable marking.

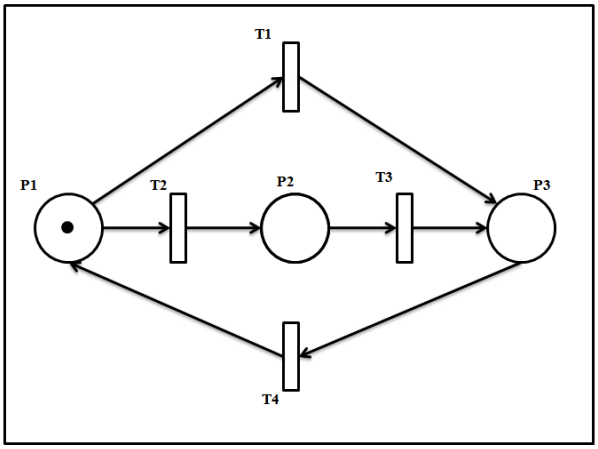

Figure 12: Petri Net model

5.3.3 ALGEBRAIC ANALYSIS

THE MATRIX-EQUATION METHOD[1]

Description: The Matrix-Equation method approaches for the study of dynamic be-

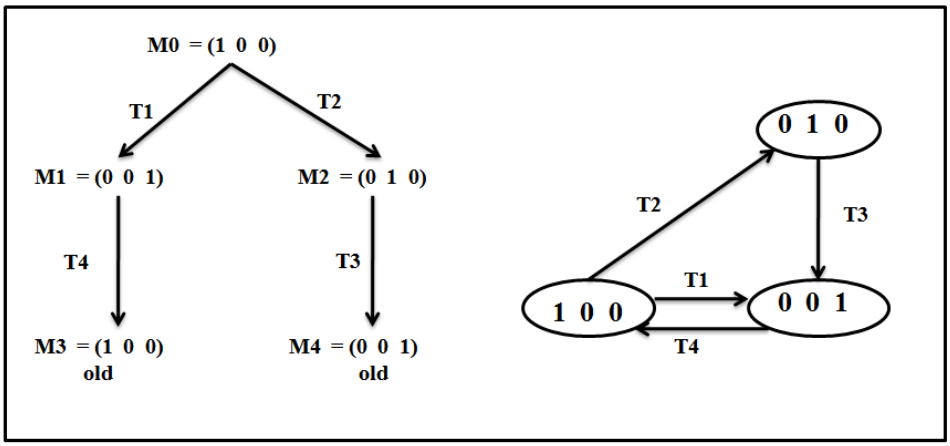

Figure 13: The Coverability tree and The Coverability graph for petri net in Figure 12

haviour of the concurrent system by Petri net model. It is an Algebraic approach to find the next state after series of firing and possible fireable states. This method is applicable only to special subclasses of Petri Nets as the approach to reachability is not always feasible.

Method: The Incidence-Matrix: For a given Petri Net model with n transition and m places, an incidence matrix A_{ij} = [a(i, j)] of n x m is formed.

$$A_{ij} = a_{ij}^{+} - a_{ij}^{-}$$
$$\text{where } a_{ij}^{+} = w[i, j] \text{ and } a_{ij}^{-} = w[j, i]$$

where w[i, j] = weight of the arc from transition i to output place j and w[j,i] = weight of the arc from input place j to transition i. [1]From the given expression it shows that, is the tokens added removed and change after the respective transition is fired for a transition to be fireable.

The State-Equation: The equation is described by Mk matrix of m x 1, where the jth entry in the matrix means the number of token in place j after kth firing and Uk matrix of n x 1 with [n-1] 0s and a non-zero entry. The ith entry of this non-zero entry is the number of times the transition i is being fired in the sequence. The reachable condition:

$$M_k = M_{k-1} + A^T U_k$$

Suppose Md is reachable from M0 after a series of firing, where d is the total number of transition fired in the series, then Md is represented as,

$$M_d = M_o + A^T{}_k \sum_1^d U_k$$

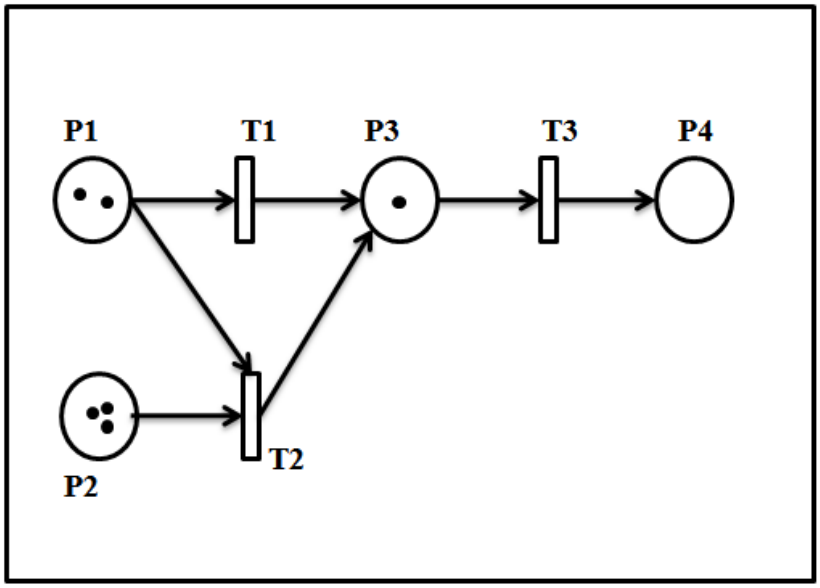

Figure 14: A Petri Net model

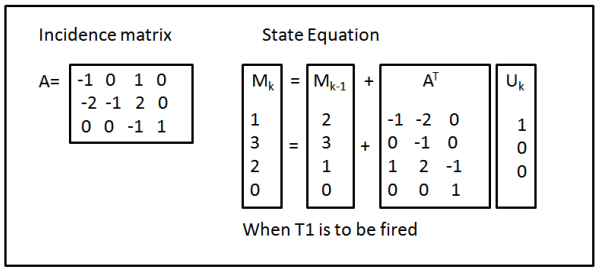

Figure 15: Incidence-Matrix and state-equation calculation for the given petri net in figure 14

6 DINING PHILOSOPHER PROBLEM

The Dining Philosophers Problem is a classic Synchronization problem of the software system. In this system five philosophers are dining; they are provided with five chopsticks, they can either think or eat. The problem symbolises working of resource management in a system. The chopstick stand for the resource and the philosopher stands for the process. Availability of the chopstick denotes the availability of the resource for the particular process to execute. The synchronizing problem helps in preventing deadlock situation in dynamic resource allocation.

In the Figure16: [2]

a. A, B, C, D, E represents the philosophers.

b. The subscript representation, p–Picks up forks, r–returns the forks, t–philosopher is thinking, e–philosopher is eating respectively.

c. a0 to a4 denotes that fork is available for the philosophers.

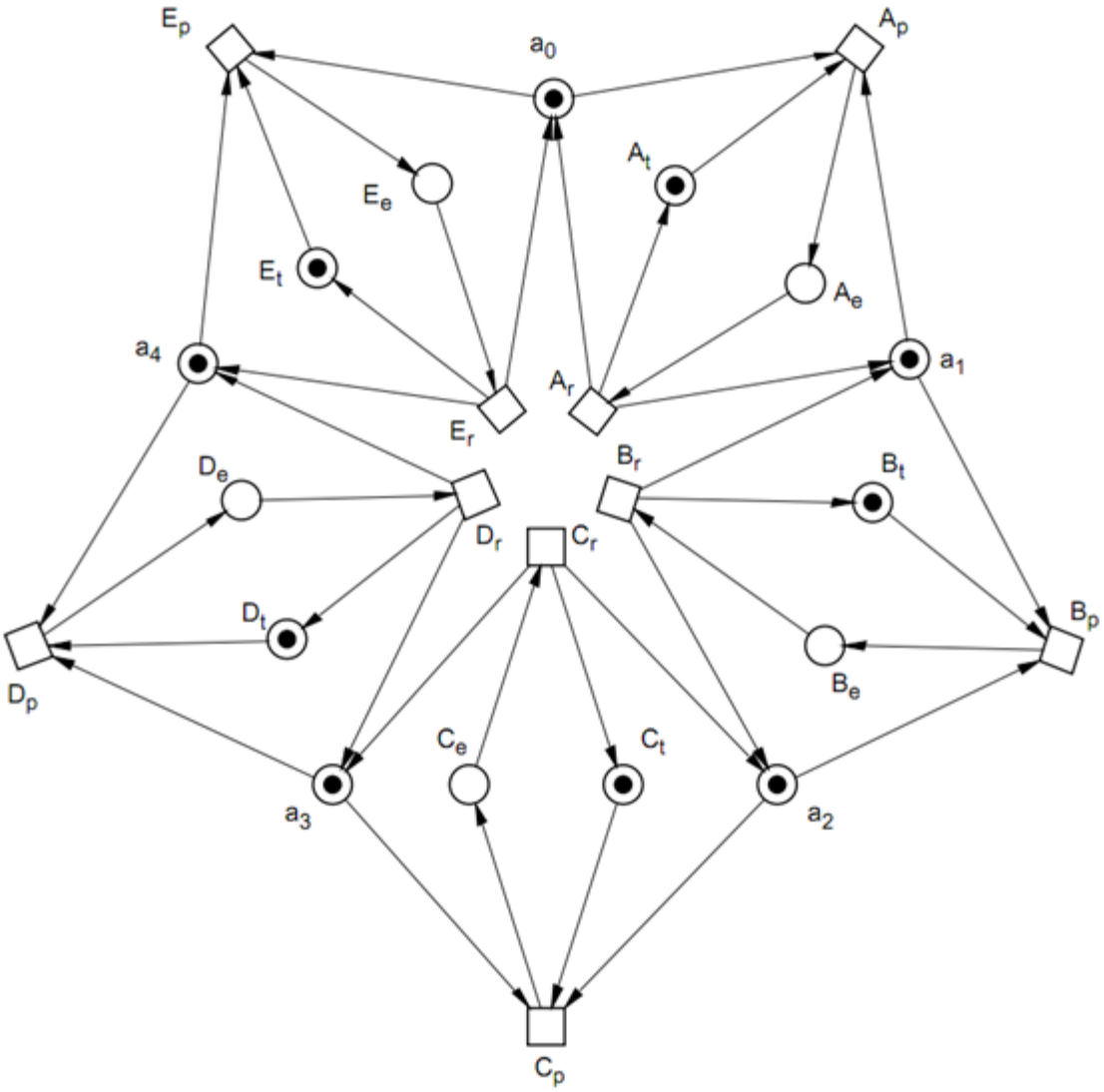

Figure 16: Petri Net model for dining Philosopher Problem

7 IMPLEMENTATION

With the help of JAVA NetBeans IDE 6.9.1, two simulators were created. The first simulator dynamically takes input when the graph is drawn. Given an option Enter the id of the transition number to be fired, when the transition number is entered, if it is fireable, shows the next state or shows an error message that the transition is not fireable. The second simulator is specially designed for Dining Philosophers Problem with five philosophers and given time constraint for each philosopher. The working of the simulator is discussed ahead.

7.1 MULTI-PROCESS SYSTEM

7.1.1 ASSUMPTIONS

An example of two bus multiprocessor system is being executed by the help of Petri Net model in the simulator. [1]The model consists of five places and five transitions;

1. P0 to P5 representing active processor, free buses, processors waiting for request to be accepted, processor having access to common memories and processor queued for common memory respectively.

2. T0 to T5 representing issuing the access request, access to memory, choosing the memory that is being accessed by another processor, end of the access where no outstanding request are made and end of the access for the processor queued respectively.

3. The initial state is suppose M0 = [5 2 0 0 0].

7.1.2 SIMULATION AND RESULTS

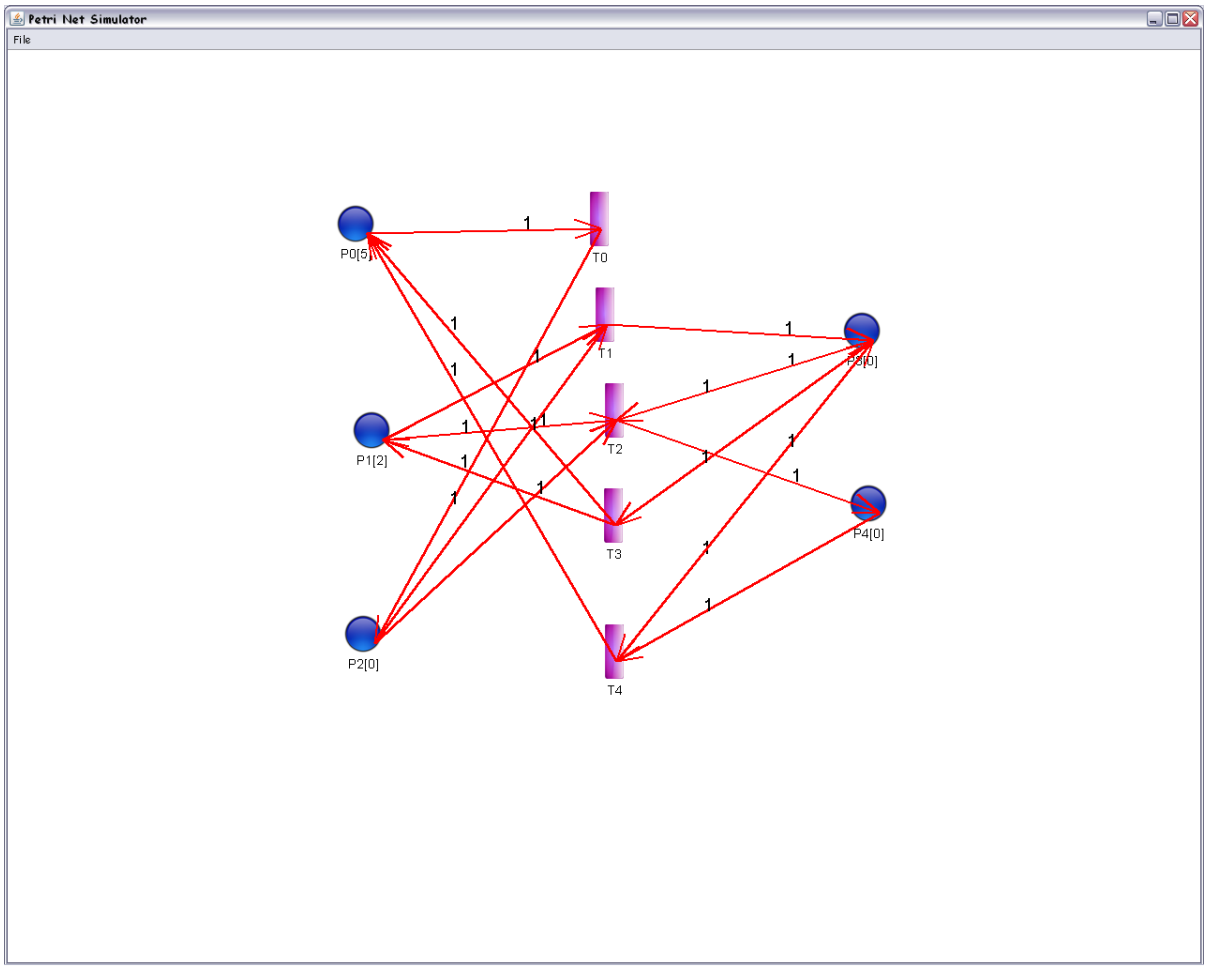

Figure 17: Screen shot of the Multi Processor. Initial state is M0 = [5 2 0 0 0]

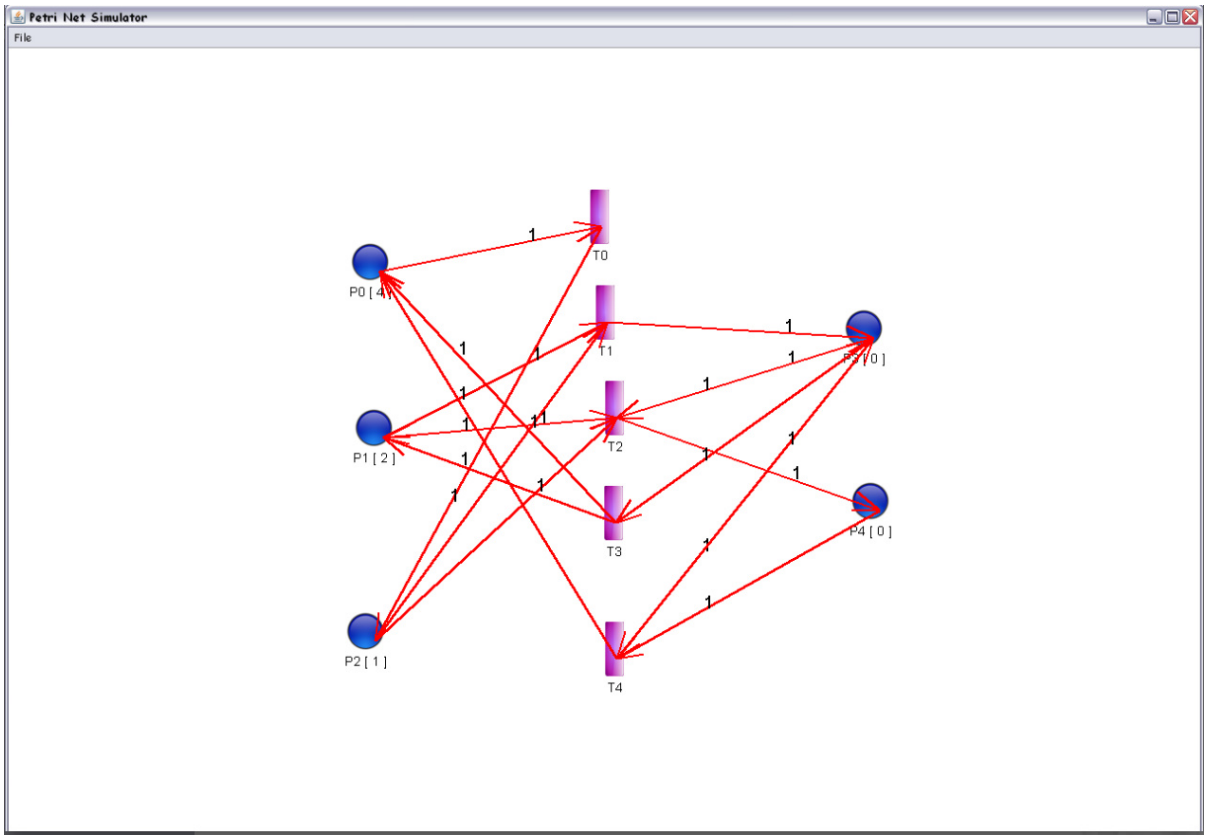

Figure 18: Firing of T0 the state changes to M1= [4 2 1 0 0]. Firing of T0 accepts request to access the memory

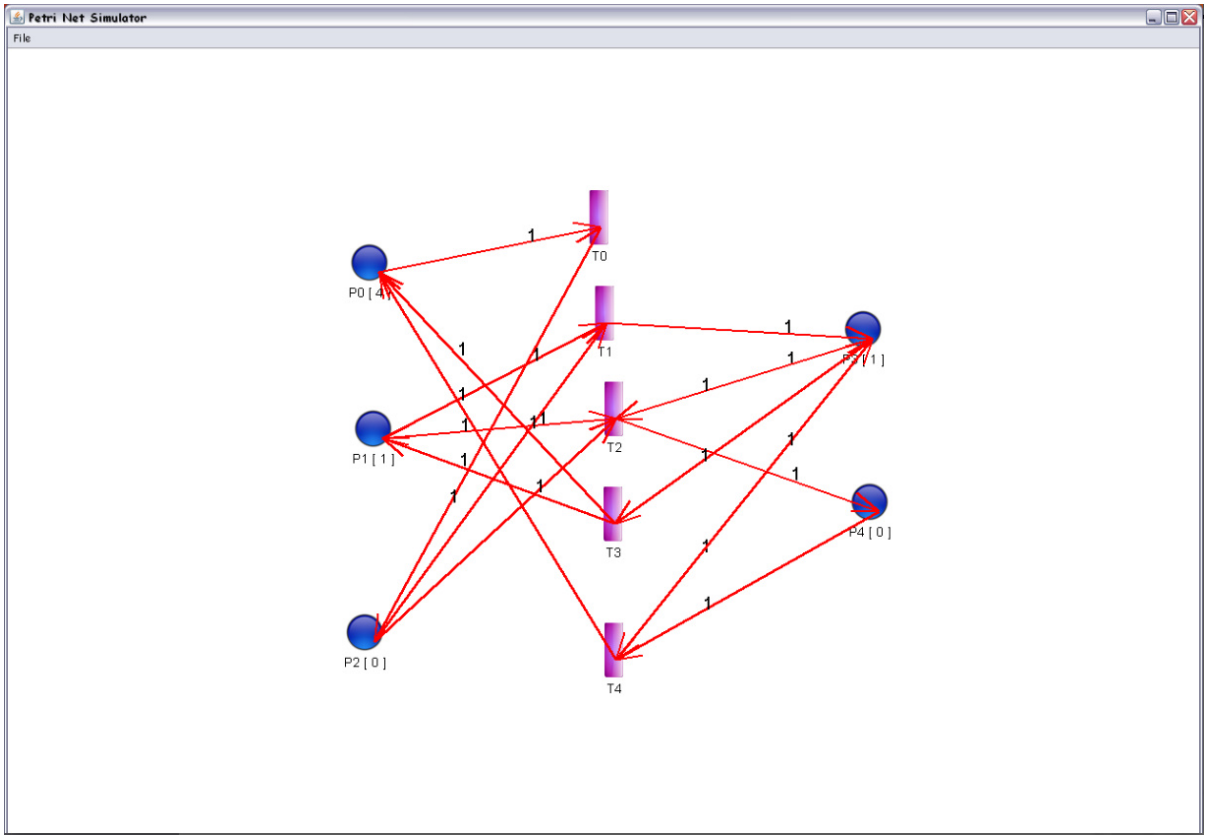

Figure 19: Firing of T1 represents the processor is accessing the memory as free bus is available,the state has changes to M2=[4 1 0 1 0]

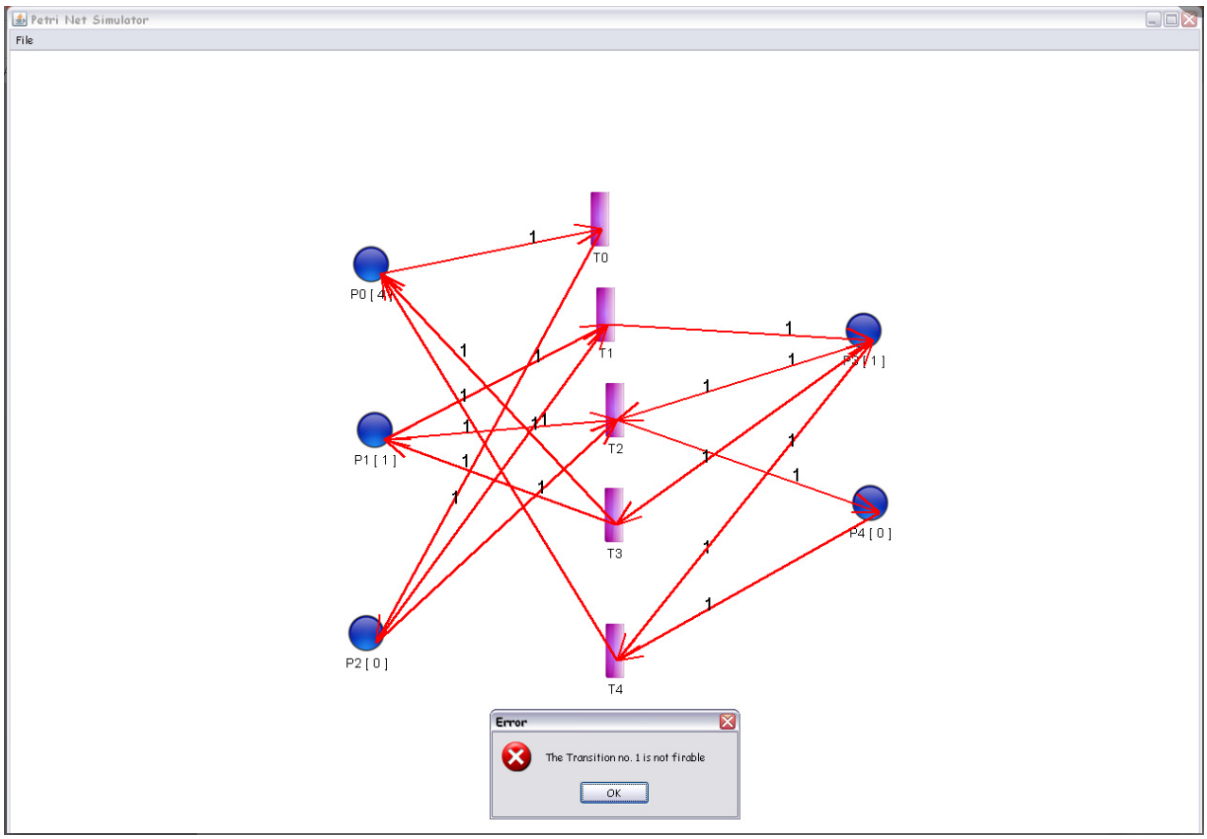

Figure 20: Firing of T1 again shows a error as no more processors request is being accessed, the state remains the same M2=[4 1 0 1 0]

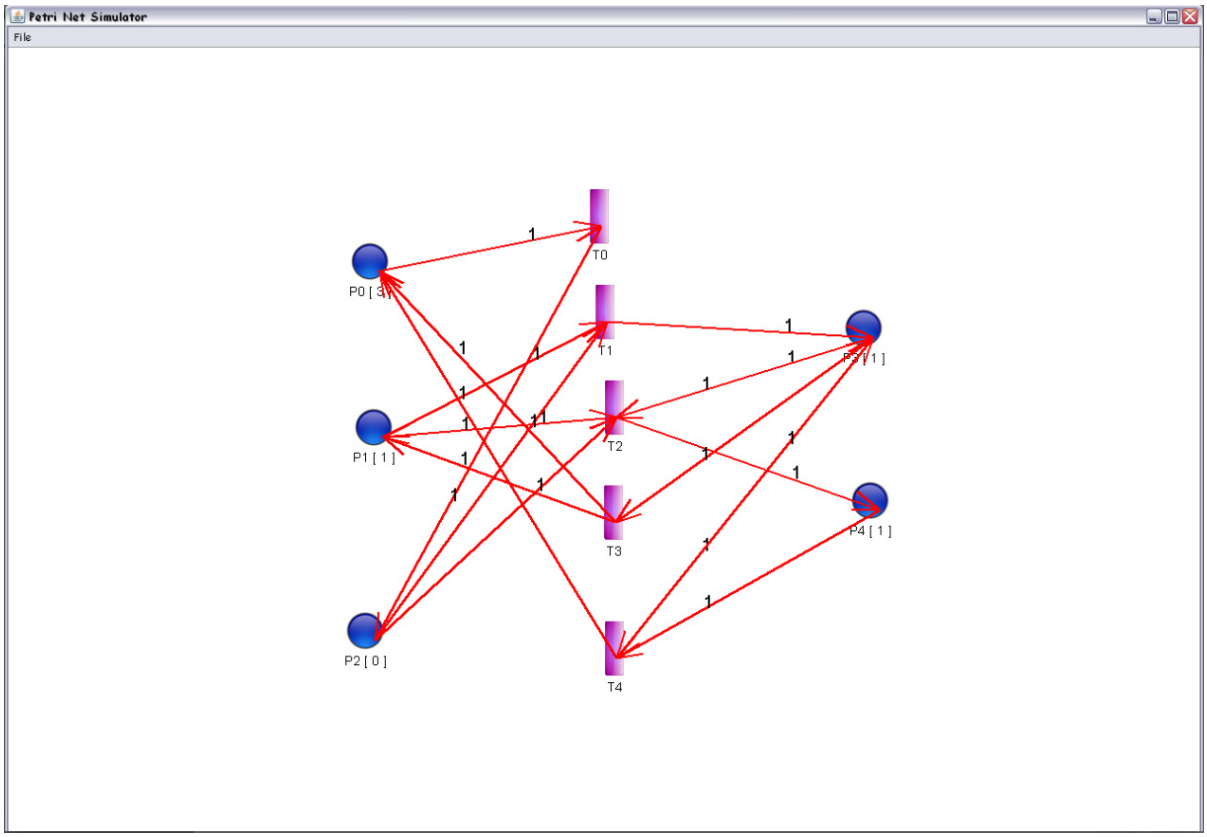

Figure 21: Firing of T0 gives access to one more processor changing to the state M3 = [3 1 1 1 0] and subsequently firing of T3 results as M4 = [3 1 0 1 1]. The processor which was queued for the access of the common memory has been grant access.

7.2 DINING PHILOSOPHER'S PROBLEM

7.2.1 ASSUMPTIONS

In the corresponding Petri Net model, 15 places and 10 transitions are used. The details and the representation are given below:

1. The representation of the initial state is M0= [P0, P1, P2,... P12, P13, P14]. i.e M0 = [1 1 1 1 1 1 1 1 1 1 0 0 0 0 0].

2. P0 to P4 represents the availability of chopsticks.

3. P5 to P9 represents the philosopher is thinking.

4. P10 to P14 represents the corresponding philosopher is eating respectively.

5. T0 to T9 are used as transitions that are to be fired.

6. T0 to T4 represents the philosophers picking up the stick.

7. T5 to T6 represents the corresponding philosophers returning the stick respectively.

8. Weighted arc that can fire one token at a time is used.

9. Time-constraint is given when T0 to T4 is fired(philosopher picks up the stick to eat).

7.2.2 SIMULATION AND RESULTS

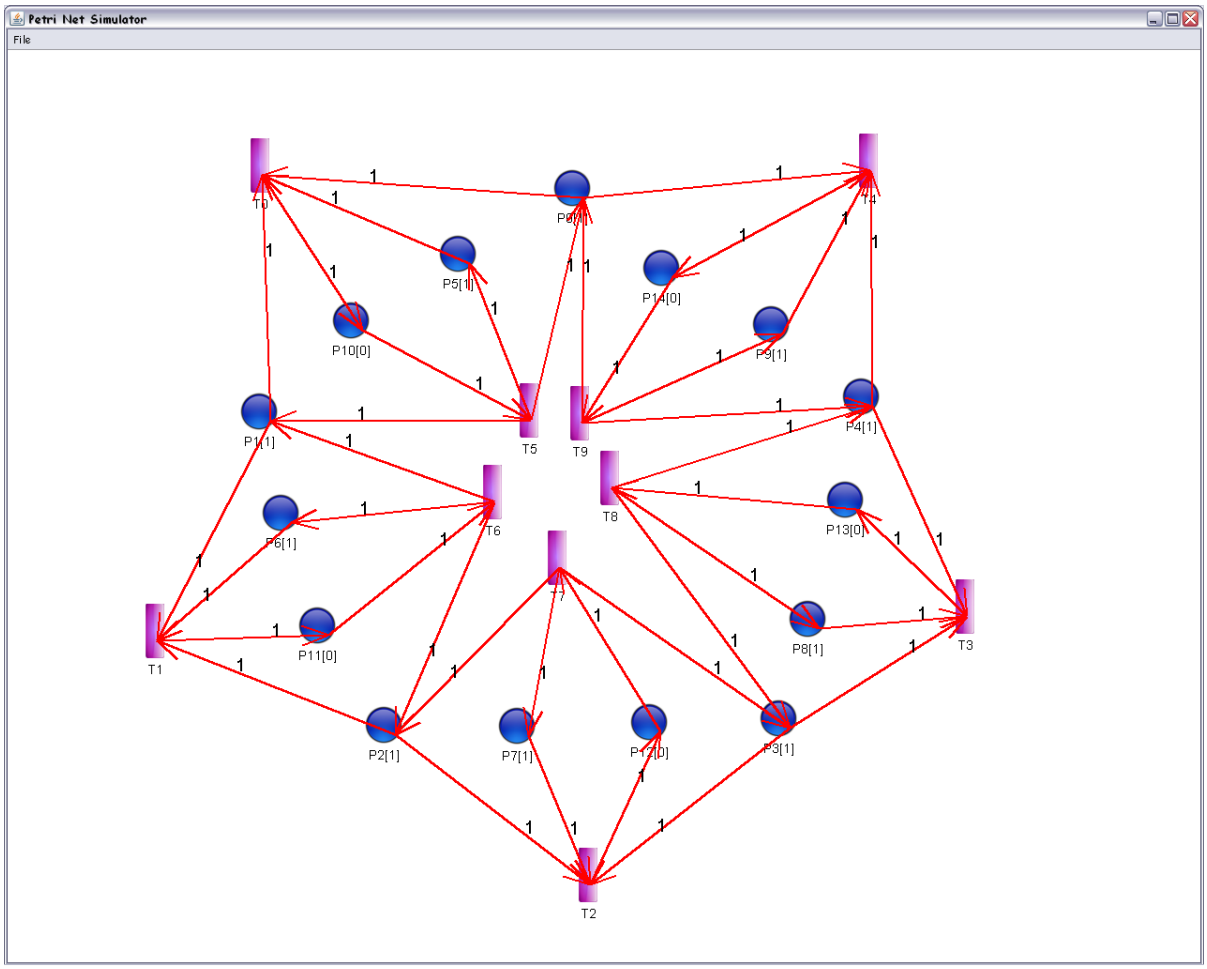

Figure 22: Initial state of the petri net model is M0 = [1 1 1 1 1 1 1 1 1 1 0 0 0 0 0].

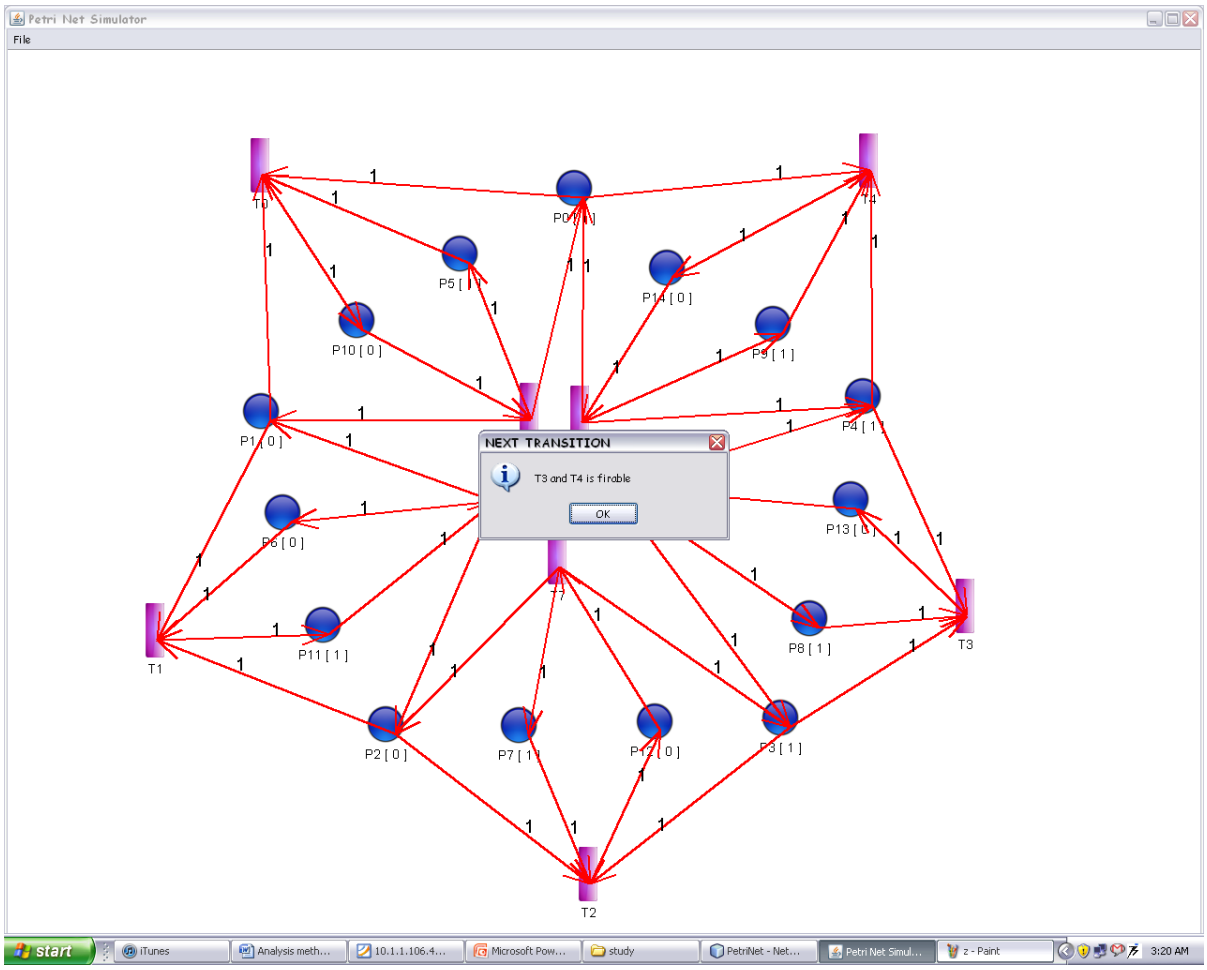

Figure 23: Suppose T1 is fired, the state changes to M1 = [10011 10111 01000]. A message T3 and T4 is fireable.

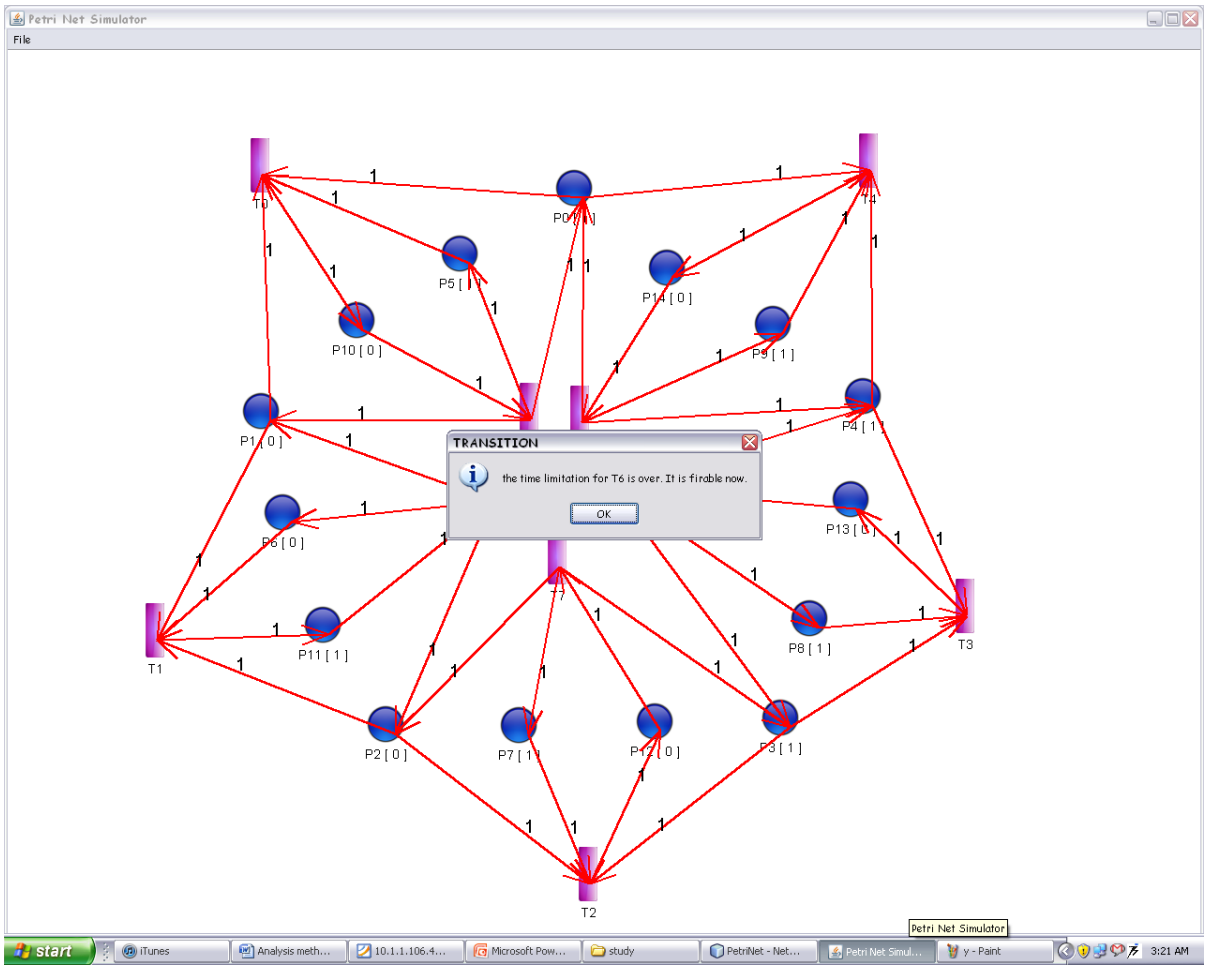

Figure 24: When T1 is fired the state changes to M1 = [10011 10111 01000] and suppose the time constraint given is 40s. A message automatically generates after a stipulated time - "the time limitation for transition T6 is over. It is fireable now". The user would come to know when will be the next transition fireable.

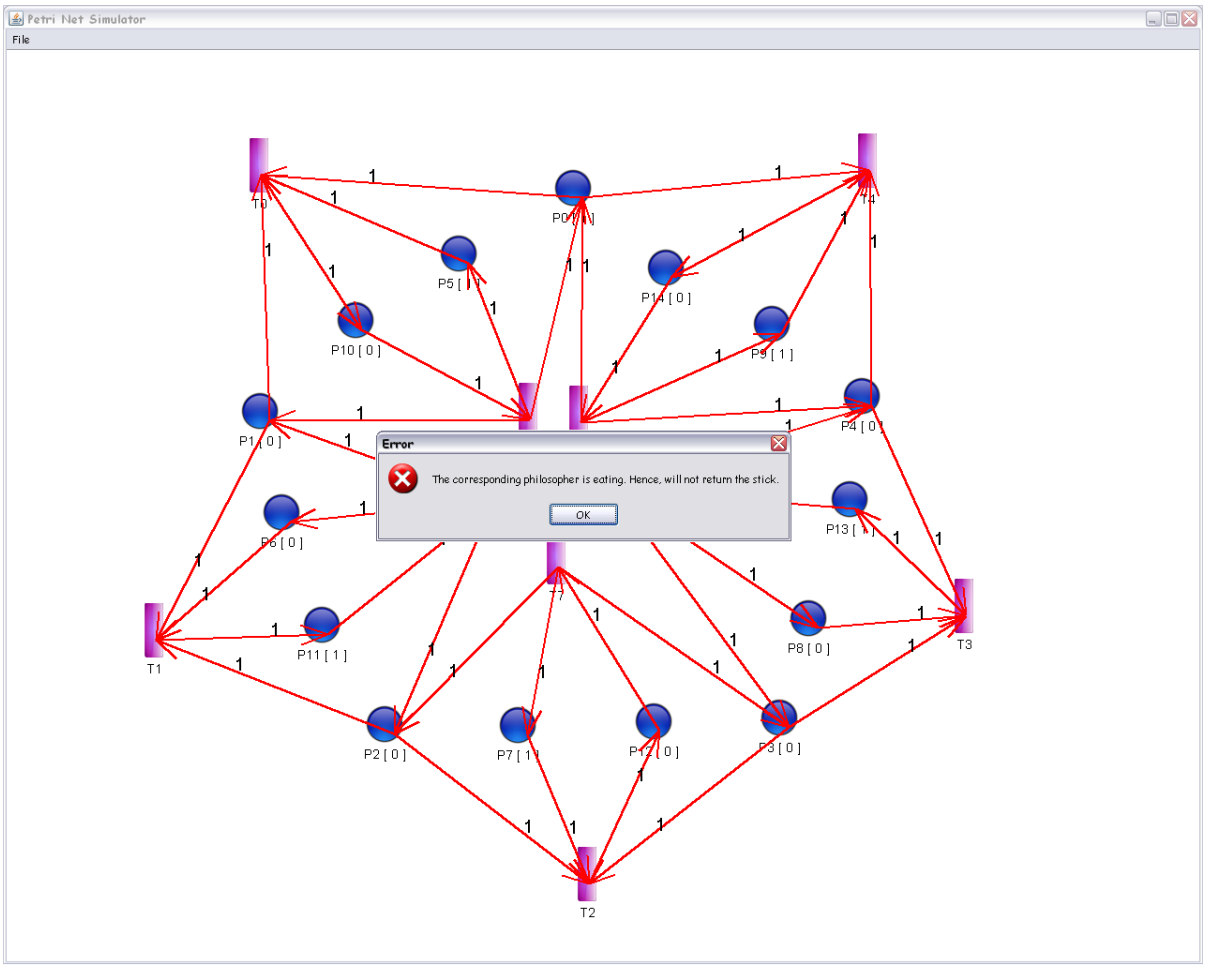

Figure 25: T1 is fired and then subsequently T3 is fired and the state changes to M2= [00001 10101 01010]. An option is given to the user for entering the time constraint and shows the next fireable states. Suppose the time-constraint given is 40 seconds and immediately firing of T1 shows an error - "T1 is recently fired "and firing of T6 or T8 generates an error message - "The philosopher is eating. Hence, will not return the stick". After stipulated time as shown in figure24 the message generates about the time limitation to be over and which are the next possible states. This implies the philosopher will not be able to pick up the stick if he is already eating.

7.3 SIMULATED SERIES OF FIRING

Transition fired	State No.	Time Constraint (in secs)	State	Message
Initial State	M0	-	[1 1 1 1 1 1 1 1 1 1 0 0 0 0 0].	
T0	M1	40	[0 0 1 1 1 0 1 1 1 1 1 0 0 0 0]	T5 is fireable is generated after 40 secs.
T5	M2	-	[1 1 1 1 1 1 1 1 1 1 0 0 0 0 0]	T1, T2, T3 and T4 are now fireable.
T1	M3	20	[1 0 0 1 1 1 0 1 1 1 0 1 0 0 0]	T3 and T4 are fireable.
T4	M4	20	[0 0 0 1 0 1 0 1 1 0 0 1 0 0 1]	T6 is fireable after x secs.
T6	M5	-	[0 1 1 1 0 1 1 1 1 0 0 0 0 0 1]	T2 and T9 are fireable.
T9	M6	-	[1 1 1 1 1 1 1 1 1 1 0 0 0 0 0]	T0, T1, T2 and T3 are fireable.
T2	M7	20	[1 1 0 0 1 1 1 0 1 1 0 0 1 0 0]	T0 and T4 are fireable.
T4	M8	10	[0 1 0 0 0 1 1 0 1 0 0 0 1 0 1]	T7 is fireable.

8 CONCLUSION

A simulator was successfully created and was user friendly. It stored the input given dynamically and could accurately calculate the next state and could show the next possible transition that are fireable. The error messages helped in preventing the user from entering the transitions that are yet not enabled. The simulator successfully helped to have deadlock-free processing multi-processor system. The synchronization was well controlled by the simulator. This type of simulator can be further created to solve the complexity of any multi processor system.

Refrences

[1] Tadao Murata, *Petri Net: Properties, Analysis and Application*, Proceedings of IEEE, Vol 77, No. 4, April 1989.

[2] Wolfgang Reisig, *The Decent Philosophers: An exercise in concurrent behaviour*, Humboldt-Univeritat zu Berlin, Institute of informatics, Unter den Linden 6 10099 Berlin, Germany, Fundamenta Informaticae 80(2007) 1-9 IOS Press.

[3] Venkatramana N. Reddy*, Michael L. Mavrovouniotis, and Michael N. Liebmant+, *Petri Net representation in Metabolic Pathways*,*Chemical Engineering Department and Institute for Systems Research, A.V. Williams Bldg. (115), University of Maryland, College Park, MD 20742, U.S.A. and +Amoco Technology Company, Amoco Research Center, MC F-2, Naperville, IL 60566, U.S.A. Proceedings from ISMB-93: 328-336(1993)

[4] Hsu-Chun Yen, *Introduction to Petri Net Theory*, Dept. of Electrical Engineering, National Taiwan University, Taipei, Taiwan, R.O.C., (yen@cc.ee.ntu.edu.tw). Supported by NSC Grant 94-2213-E-002-087, Taiwan

[5] K.M. Chandy and J. Mishra, *The dining Philosophers problem*, University of Texas at Austin. ACM Transaction on Programming Language and System Vol. 6, No.4 October 1984.

[6] Wilfried Brauer, Wolfgang Reisig,*Carl Adam Petri and "Petri Nets"*. A paper translated from Informatik - Spektrum, Vol. 29, Nr. 5, pp 369-374, Springer Verlag, 2006.

[7]$http://www.informatik.uni-hamburg.de/TGI/PetriNets/applications, Uni$

[8]$http://torguet.free.fr/java/Petri.html, Patrice's Sure Sites, Petri Net Simulator.$

[9]$http://crypto.cs.mcgill.ca/\ jourdain/projects/petri/petri.html, A Petri Net Simulator.$