International Journal of Computer Information Systems and Industrial Management Applications (IJCISIM)
ISSN: 2150-7988 Vol.2 (2010), pp.069-076
http://www.mirlabs.org/ijcisim

Transaction Management for Distributed Database using Petri Nets

Bidyut Biman Sarkar [*1], Nabendu Chaki[*2]
[*1]*Techno India, Salt lake, Kolkata, Inda*
[*2] *University of Calcutta, Kolkata, India*
[*1]*bidyutbiman@gmail.com,* [*2]*nabendu@ieee.org*

Abstract

Large and medium sized organizations are functionally distributed in a structured or unstructured form over different locations like continents, countries, plants, divisions, departments, laboratories, work-groups and so on. As there is no single global clock available for synchronizing the transactions, some cost effective yet easily deployable generic framework for asynchronous transmission is necessary. The transaction management mechanism for the distributed environment must ensure that the sequence of updates is safe and reliable when committed on the stable storages at different locations. In this paper the Two Phase Commit (2PC) protocol for distributed transactions is modeled with the help of a timed Petri net to analyze the ACID property for consistent commitment of distributed transactions.

Key Words: Distributed transactions, 2-Phase Commit, Timed Petri Net, Time Petri Net, Virtual Data Warehouse

1. Introduction

The work proposed in this paper is an extension of [1]. Transaction management in a distributed environment has been analyzed by introducing a sequence diagram in this extension. Some of the key issues for successful commit and failure of distributed transactions are described in section 3.1. The Reachability analysis is performed and included in this extension in section 4.2.1. More number of contemporary literatures is reviewed; citations of recent relevant work on Petri Net models are presented.

In a distributed environment communication between objects or entities belongs to multiple servers is made through message passing. Due to non availability of centralized clock the synchronization process depends on the time at which inputs are received, messages are lost in transit and the speed of the processes. Complexity of the transaction is directly proportional to the number of operational locations. As for example; if the quality control department operates as a separate cost centre or export processing needs to setup at a new location. In such situations the number of sites will increase. Cyclomatic complexity will increase, time complexity for distributed query processing will also increase and so as the chances of resiliencies will also increase [2]. Message passing operations can be used to construct protocols to support particular process role and communication pattern [3]. As for example in a business to business communication in an e-commerce application, a clearing house (financial entity) as a third party is always present to record all financial transactions. The transaction times at the computer systems of the business entities and the financial institutions must tally with the allowed time delays. Therefore the designing of a communication protocol for a distributed transaction processing system is considered to be one of the most critical tasks.

A brief review on four recent different contemporary works on distributed decision support systems has been carried out in [4]. Most of these are web-based and E-Commerce applications. Interoperability among the distributed systems is the key issue in the present paper. Some of the architectures that are used for integration of distributed legacy systems are CORBA, Java J2EE, XML, SOAP, DCOM, Java RMI, EAI, and OMG, MDA. CORBA allows applications to communicate with one another efficiently and the Extensible Mark up Language (XML) is used to process data on the WEB. The Distributed Component object Model (DCOM) is a protocol that enables components of the architecture to communicate directly over a network [5].

Petri Net [4] is one of the most widely used modeling and analysis tool. The classical Petri net is a directed bipartite graph .The model describes the states, events, conditions, choice, iterations and parallelism. The two types of nodes are called places and transitions. Places and transitions are connected via arcs. Places are graphically represented by circles, transitions by bars. Places can store tokens, represented by black dots. A distribution of tokens on the places of a net is called a marking, and corresponds to the ``state" of the Petri net. A transition of a net is enabled at a marking if all its input places contain at least one token. An enabled transition removes one token from each of the input places, and adds

one token to each of its output places. This is called the firing rule. The formal definition of a Petri Net is described by four-tuple and can be presented as:

PN= (P, T, D-, D+) (1)

Where, P is the set of places and |P|=m

T is the set of transitions and |T|=n

D^-:P ⊗ T→N is the pre incidence matrix that specifies the arcs directed from places to transitions.

D^+:T ⊗ P →N is the post incidence matrix that specifies the arcs directed from transitions to places.

D(P,T) token changes in place P_i for transition T_i and the P-invariant X indicates the conservation of tokens if X ⊗ D(P,T)=0 and T-invariant Y indicates the system stability and steady state, if Y ⊗ D^-(P,T)=0

As for example, let us consider a primitive conveyor belt system used in any process industry with three places: P_1–initial, P_2–current and P_3 – break down and four transitions t_1, t_2, t_3, and t_4 describes as follows:

t_1 – task starts, t_2 – task complete, t_3- task down, t_4- task maintenance.

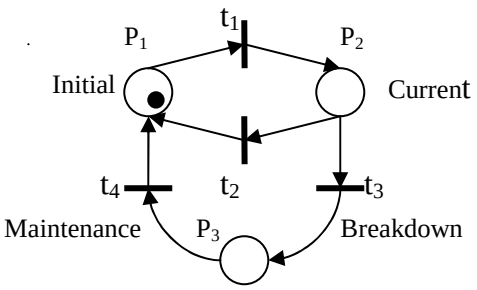

Figure 1: Conveyor Belt in operation

The markings at the respective locations are:

	P_1	P_2	P_3
Initial (Idle)	[1	0	0]^T
Current (working)	[0	1	0]^T
Break Down	[0	0	1]^T

P-invariants indicate token conservation and T-invariants represents system stability and steady state of the system.

P-invariants

P_1	P_2	P_3
1	1	1

The net is covered by positive P-invariants, therefore it is bounded and the P-invariant equation may be represented as M (P_1) + M (P_2) + M (P_3) =1, Where, M is the marking at the respective place.

T-invariants

t_1	t_2	t_3	t_4
1	1	0	0
1	0	1	1

The net is covered by positive T-invariants; therefore the net is bounded and live.

This paper is organized in five sections with relevant subsections. Section 2 provides the basic specification on time Petri Nets. Section 2.1 presents the timing dynamics of a Timed Petri Net. Section 2.2 depicts the timing dynamics of a Time Petri Net. Section 3 describes the traditional distributed transaction processing intricacies along with the web based distributed transactions. Section 3.1 presents the 2PC-commit protocol for distributed systems with the help of an USE CASE diagram and in section 3.2 a sequence diagram of the 2PC Distributed System is described. Section 4 represents the Petri Net model of the 2PC commit protocol. Section 4.1 describes the Incidence Matrices of the Petri Net model. Section 4.2 deal with the Reachibility Graph of the Petri Net model and in section 4.2.1 the Reachability Analysis of the model is performed and presented. Section 5 indicates future scope of the work plan.

2. Time Petri Nets

Adding the timing parameter with basic Petri Net models enable us to simulate interesting real time systems. One such application of spiking neural P-system [15] is a deterministic SN-P model where communication among the neurons used to take place through electrical signals of identical voltage called spikes. Synapses are used as links with the neighbouring neurons.

There could be various temporal constraints in modeling such systems. The constraints could be a fixed or a variable type. When the transition is a fixed type constraint with a single time delay is known as Timed Petri Net and it observes a strong firing mode. As for performance evaluation of a timed Petri Net model the time to fire and enabled a particular transition the firing rules may be further improvised and more constraints may be added. Instead of considering a single time delay, when a time domain is used and observes a strong firing mode is renamed as Time Petri Net [6].

In order to model the temporal aspect of concurrent and distributed applications the addition of delay time will restricts the dynamic behavior of the net. But by adding color to the tokens will help in identifying the objects uniquely and finally if an hierarchy is added then decomposition of a complex system becomes easier. So, when all these three features time, color and hierarchy are added to a basic Petri Net model the Petri Net Model becomes High Level Petri Net [7].

Timed workflow graphs are used for modeling the temporal aspects in a Work Flow Management System (WFMS), where first order predicate logic is used for modeling, specifying and analyzing the temporal issues at design and execution times[8]. In a Time Petri Net model

time is associated to transitions. Transitions represent activities and activities take time. Timed Petri nets are similar to Petri nets with the addition of a clock structure associated with each timed transition. Let us now define the time Petri net as a five tuple:

$TPN = (P, T, D^-, D^+, V)$ (2)

A timed transition t_j once it becomes enabled fires after a delay v_{jk}. Time delays are of deterministic type, non deterministic and stochastic types. Deterministic delays allow for simple analysis methods with limited applicability. The models handling nondeterministic delays use time intervals to specify the duration of the delay. In these models each delay is described by a probability distribution function. Formally the Timed Petri Nets allow strong firing mode, i.e, a transition, t_j with a delayed time, T_{del}, will immediately fire at time when necessary tokens have arrived. During the time period from T_0 to $(T_0 + T_{del})$ the tokens are preserved for t_j and consequently no other transitions can use those tokens. At time $(T_0 + T_{del})$, the tokens must be removed from t_js input to output places.

2.1. Timing Dynamics of a Timed Petri Net

Given the current enabled state x, the following pseudo code shows how to evaluate the next enabled state x' in a Timed Petri Net [6]. Let us assume that;
x is the current enabled state
e is the transition that caused the PN into state x
t is the time that the corresponding event occurred
e' is the next transition to fire (firing transition)
t' is the next time the transition fires($t'=t+t_{del}$)
x' is the next state given by x' = f(x, e') where, f() is the state transition where function.

2.2. Timing Dynamics of a Time Petri Net

In a Time Petri Net two times are associated with each transition [9]. Smallest and the largest of these times for any transition are marked as early-finish-time (EFT) and late-finish-time (LFT) respectively. The firing Interval of the transition is the difference between EFT and LFT. States are pairs x = (M, I) in which M is a marking and I is a firing Interval function. Firing a transition t, at time Ω from a state x = (M, I), is allowed if both the following conditions hold:
(i) The transition is enabled;
(ii) Time Ω is comprised between the EFT and the smallest of the LFTs among the enabled transitions. On firing t at time Ω from a state x= (M, I) moves to a state x' = (M', I') and is computed as follows:
1) The new marking M' for each place is defined for any place P, as in Petri Nets, as:
M'(P) = M(P) — Previous(t, P) + Next(t, P)
2) The new firing intervals I' for transitions are computed as follows:
a) For all transitions not enabled by the new marking x', then empty;
b) For all transitions e enabled by marking M and not in conflict with t, then $\max(0, EFT_e - \Omega)$, $LFT_e - \Omega$, where EFT_e and LFT_e are the lower and upper bounds of interval I for transition e, respectively;
c) All other transitions have their interval set to their Static Firing Interval.

3. Distributed Transaction Processing

In a distributed transaction at least two or more network hosts are involved. Net work hosts provide Transactional resources while the transaction manager/coordinator is responsible for creating and managing a global transaction. The transactions can be flat or nested. Commit or abort transaction depends on agreement of all the participating servers and two phase commit protocol is used. Concurrent transactions must observe locking mechanism or timestamp protocol or optimistic concurrency control protocol. The global transactions may encounter multiple resiliencies like failure of hosts, failure of the network connection or deadlock occurs and finally the global transaction is responsible for recovery of the aborted transactions. Internet and web-based distributed transactions are interoperable with the traditional distributed transaction processing systems; Let us now summarize the characteristics of the distributed transactions

- Transactions are referred to all discreet tasks that must be performed as a unit to accomplish a goal.
- Transactions may involve tasks that are done by one or more participant.
- Transactions make sense when perform in conjugate with some other tasks.
- Transactions must maintain the ACID properties (Atomicity, Consistency, Isolation and Durability).
- In a conventional distributed environment, transactions are short lived and resources are locked over a specific duration and the participants will act under a transaction manager.
- Distributed transactions under Internet and Web services are of long durations.
- Participants may not allow their resources to be reserved for long durations. Reservation is a characteristic of the Isolation property of ACID.

WEB based distributed communication infrastructure between the participants are not always reliable. It depends on the communication network standard. Any Web-based transaction may need to succeed even if only some of the participants choose to confirm the transaction and others cancels it. All activities in a distributed

environment are logged. Transactions that have to be undone perform compensation to return to previous state.

3.1. 2-Phase Commit Protocol for Distributed System

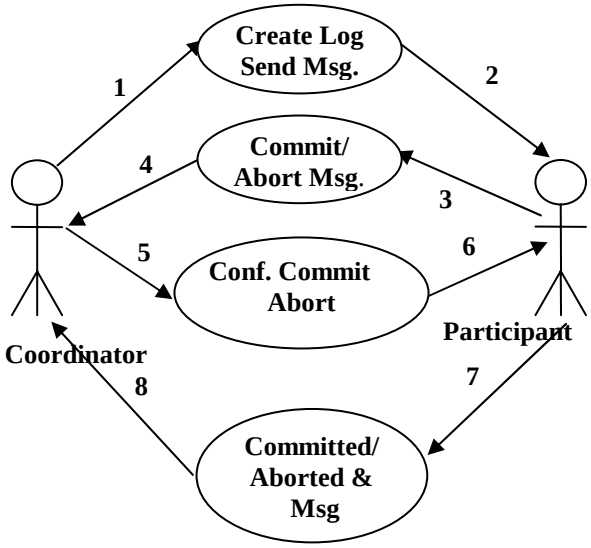

Figure 2: Use Case Diagram

Success of 2-Phase Commit protocol [10] of distributed transactions primarily ensures that all the distinct servers must be agreed either to commit or to abort. The initiator process is called coordinator and all others are identified as participants. The coordinator first communicates a message of commit/abort to all participants and waits for a fixed duration. After a fixed interval of time repeat requests are made to all non responding participants. The process continues over a fixed interval till all responses are acknowledged. To maintain the ACID property of the participating transactions, the server can't abort part of a transaction or commit without getting the acknowledgement from all participants. However, there can be situations when transaction gets aborted at some point of execution when the server is crashed or deadlock is detected.

Let us now model the coordinator, participant activity of 2-phase commit protocol with the help of a use-case diagram (figure 2). The model consists of two distinct entities namely coordinator and participant. The occurrences of events are taking place between these two entities. There are four events in the process (create log, send message), (Commit/Abort message), (confirm-commit, confirm-abort) and (Committed/Aborted, send message) .The events are sequenced here numerically to describe the order of the occurrences. A class diagram is presented in (figure 3). The pseudo code of the use-case functions following the timing sequence is as follows:

1: A LOG record is created at the stable storage with the coordinator marked as "prepare".
2: MSG: Message to participants to "create" and activate time out.
3: A LOG is created at the participant's stable storage and marked as "prepare. Write "Ready" or "Abort" message to the log at the participant's log.
4: MSG: Message to coordinator "Ready" or "Abort".
5: Coordinator acknowledges and checks all participants reply. Check out with time out parameter and write the decision of commit or abort in the log.
6: MSG: Send "Abort/Commit" message for confirmation to all Participants.
7: Participants acts according to the message received from "(6:)" above. Write in the log. Execute the intended operation.
8: MSG: Send the acknowledgement message to all Coordinators.
9: On receipt of all acknowledgements from the participants write "complete" in the coordinators log.

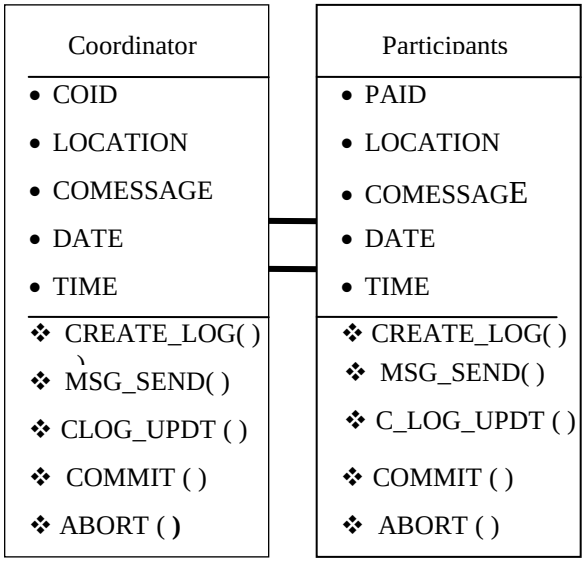

Fig 3: Class Diagram of 2PC

3.2 Sequence Diagram of the 2PC Distributed System

There are six different states of activities listed in sequence as follows:
1→ Coordinator
2→Create Log and send MSG
3→ Commit message / Abort message
4→ Confirm Commit/ Abort process
5→ Committed / Aborted & send Message
6→ Participants

The time domain is from [t_1, t_2, t_3, ----------, t_9]. The

Coordinator→1 represents the time intervals of the four activities (2 to 5). Dashed lines indicates the virtual lines, where from the activity should start at some later instant of time. Here t_1 is the starting time when the coordinator creates a log record at the stable storage and at time $t_1 + \omega = t_2$ the message reaches to the participants.

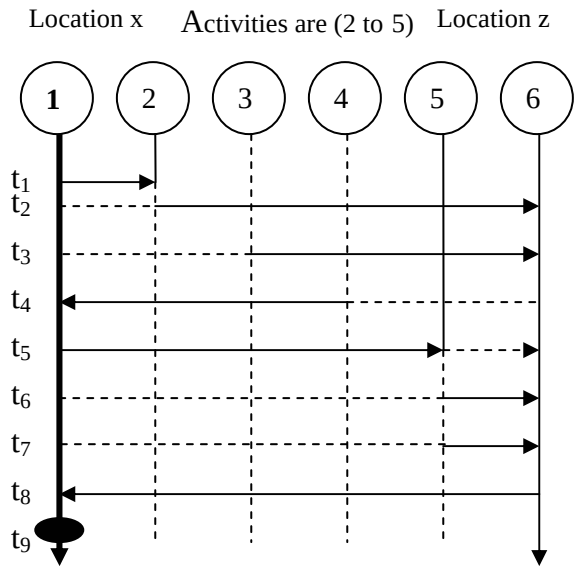

Figure 4: Time Sequence Diagram of 2PC

In the similar fashion the activities are executed till it reaches to t_8. At this point all participants confirm their activity and at $t_8 + \omega = t_9$ the activity will be completed. The following pseudo code represents the basic algorithm of a timed execution sequence of 2PC protocol.

```
Begin
Activity ← initial t1
Clock=0
Repeat
  For j= t1 to t9  do
   Begin
    If {tk is enabled} then
      <Perform the corresponding activity>
        ω = ω + clock -time
        until Ωk <= ω  [Ωk predefined time value]
   End
```

If all the activities are performed within time Ω_k then the distributed 2PC commit process can be performed successfully otherwise a time out condition will be activated. There may be various reasons for possible time outs. Let us indicate some of the possible reasons due to which distributed transactions fails to commit are:

- Deadlock Occurs
- System site failures
- Processor failure
- Media failures
- Power supply failure
- Main memory failure
- Main memory contents are lost, but secondary storage contents are safe
- Secondary storage devices Failure
- Head crash / controller failure
- Communication failures
- Lost Messages or undeliverable messages due to Network partitioning

4. Petri Net Model for 2PC Protocol

A brief overview of the distributed frame work and its 2-Phase Commit functions are presented in section 3.1. There are two different locations: coordinator location and participants location distributed geographically and marked with doted vertical line in (figure: 5) of the Petri Net model consisting of nine places and eight transitions. Five places are marked for coordinator and four places for participants.

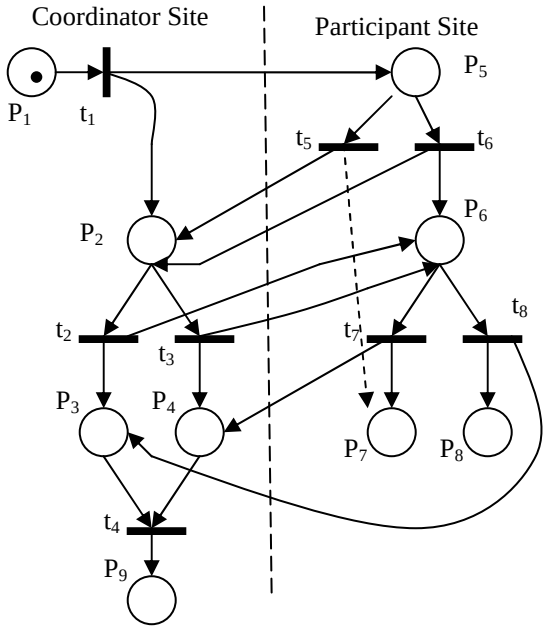

Figure 5: Petri Net Model of 2PC

Table 1: Place & Transition Function of Coordinator

Places		Transition function	
P_1	DTM of coordinator	t_1	Message to DTM of the participating site
P_2	Wait state for Commit /Abort	t_2	i)Commit at Global Log ii)Commit Msg to participants
P_3	Global Commit	t_3	i)Abort at Global Log ii)Abort Msg: to participants
P_4	Global Abort	t_4	i)Commit or abort Global Log ii)Write Complete to Global log
P_9	Complete		

Table-1 and Table-2 represents the place and transition functions of the coordinator and the participant sites. DTM is the transaction manager at the coordinator side responsible for begin the transaction and it is enabled at P1 when there is at least one token present.

Table 2: Place & Transition Function of Participant

Places		**Transition function**	
P_5	DTM of participant	t_5	Abort message to local log and coordinator to wait state (P_2).
P_6	Ready state for Commit	t_6	i)Commit Msg. at local Log ii) "Commit" Msg to coordinator wait state (P_2).
P_7	Abort state	t_7	i)"Abort" Msg: to Coordinator ii)Abort transaction
P_8	Commit state	t_8	i)Commit at local Log ii) Commit transaction iii)MSG: to Global log

4.1. Incidence Matrices of the Petri Net Model

There is a pre-incidence matrix (3A) representing the initial state, Post-incidence matrix (3B) representing operational state and the combined matrix (3C) representing the overall location wise state at any specific transition of a location called marking. Each marking can be used to analyze the Reachability of the Net Model.

In table 3(A) the row constituents P_1, P_2, P_3, P_4 and P_9 representing the coordinator locations and P_5, P_6, P_7, P_8 are the participants locations. Columns t_1-t_8 are representing the transitions. The pre-incidence matrix is represented by $[D^-]$ and the token status at any $[p,t]_{ij}$, where I={1,2,…..9} and j={1,2,…8} before commencement of the process is presented.

Table-3(A): Pre Incidence Matrix

Transition/Loc-Coord	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
P_1	1	0	0	0	0	0	0	0
P_2	0	1	1	0	0	0	0	0
P_3	0	0	0	1	0	0	0	0
P_4	0	0	0	1	0	0	0	0
P_9	0	0	0	0	0	0	0	0
Loc(Participant)								
P_5	0	0	0	0	1	1	0	0
P_6	0	0	0	0	0	0	1	1
P_7	0	0	0	0	0	0	0	0
P_8	0	0	0	0	0	0	0	0

The post incidence Matrix is represented by $[D^+]$ and the table 3(B) shows the token distribution of row constituents P_1, P_2, P_3, P_4 and P_9 of coordinator locations and P_5, P_6, P_7, P_8 of participant's locations. The column constituent's t_1-t_8 is representing the transitions after enabling the process.

Table -3(B): Post Incidence Matrix

Transition/Loc-Coord	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
P_1	0	0	0	0	0	0	0	0
P_2	1	0	0	0	1	1	0	0
P_3	0	1	0	0	0	0	0	1
P_4	0	0	1	0	0	0	1	0
P_9	0	0	0	1	0	0	0	0
Loc(Participant)								
P_5	1	0	0	0	0	0	0	0
P_6	0	1	1	0	0	1	0	0
P_7	0	0	0	0	1	0	1	0
P_8	0	0	0	0	0	0	0	1

Table-3(C): Combined Incidence Matrix

Transition/ Loc-Coord	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
P_1	-1	0	0	0	0	0	0	0
P_2	1	-1	-1	0	1	1	0	0
P_3	0	1	0	-1	0	0	0	1
P_4	0	0	1	-1	0	0	1	0
P_9	0	0	0	1	0	0	0	0
Loc(Participant)								
P_5	1	0	0	0	-1	-1	0	0
P_6	0	1	1	0	0	1	-1	-1
P_7	0	0	0	-1	1	0	1	0
P_8	0	0	0	-1	0	0	0	0

The combined incidence matrix in table 3(C), shows the token status at any instance after initiating the process. The combined matrix is computed as $[D^+ – D^-]$.

Places are represented as: $[P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8, P_9]$

Transactions represents: $[t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8]$

Initial marking M0 is: $[1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T$

P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8 and P_9 none of them are covered and hence the net is not covered by P-invariants. The same is the case for the transitions t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8 and the net is not covered by T-invariants.

4.2 Reachability Graph of the PN Model

The Reachable place of a Petri Net can be expressed by the Reachability graph [11], which is a directed graph and the nodes of the graph are identified as markings of the Petri Net $R(N, M_0)$,where M_0 is the initial marking and the arcs are represented by the transitions of N. The graph is used to define a given Petri Net N and marking M, whether M belongs to R(N). Each initial marking M_0 has an associated Reachability set .This set consists of all the markings that can be reached from M_0 through the firing of one or more transitions. In our case the reachability graph starts with initial marking M_0.

Reachability analysis is a basic dynamic property of any system. Firing rule of an enabled transition changes the token distribution in a net according to the transition rule but the equality problem is still undecided [12].
M_0 = [1 0 0 0 0 0 0 0 0]^T and finally reach to state M_9 =[0 0 0 0 0 0 0 0 0 1]^T, where we conclude the session for the current transaction in process.

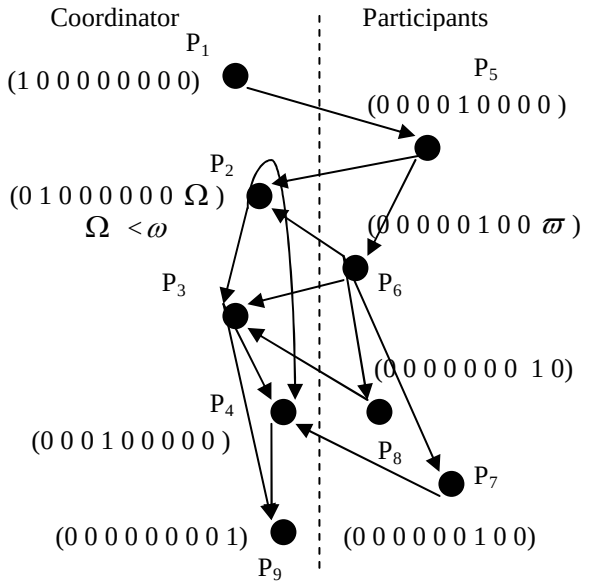

Figure 6: Reachability Graph

4.2.1. Reachability Analysis

Let us now study some of the properties and behavior of the model based on the Reachability Graph presented in Figure 6 above[13].

Safeness: Any place of a Reachability grapgh is declared safe, if the number of tokens at that place is either 0 or 1. In our case the graph clearly shows that any of the places [$\mathbf{P}_1$ _ $\mathbf{P}_9$] represents a combination of 0(no token) and 1(token), which implies that if the firing occurs there will be a token at the position bit other wise no token. Thus it shows each of the places has a maximum token count 1 or 0 and is declared safe and as all the places in the net are safe, the net as a whole can be declared safe.

Boundedness: The boundedness is a generalized property of safeness. The limitation of token numbers in a place restricted to 1 in case it is safe is enhanced to some integer i, where i is known before hand for a place or we call it as a constraint to check the overflow condition at any stage calculated once at start. Boundary value for each place will be the maximum token count for that place. When there is no overflow at any place, then the design guarantees the boundedness of the mode. In our case at each stage from [P_1- P_9], i=1 and hence it is bounded.

Conservativeness: Conservation property of a Petri Net model checks the number of tokens remains constant before and after the execution. The process is to count the sum of all tokens at their initial markings. Next the Reachability tree is traversed and the sum of all tokens is calculated for each marking in the tree. In our case it is 1. If all the markings in the Reachability tree have the same sum of tokens, then the Petri Net is declared to be strictly conservative. So our model is also strictly conservative. However, it will not be out of place to mention that in most cases due to process transformation explicit token counts are difficult to obtain to prove the conservation.

Liveness: The liveness property of a Petri Net is used to show continuous operation of the net model or in other words, it can be said that the system will not get into a deadlock state as the process of commit or abort needs to perform some transaction processing activity. The possibility of deadlock or live lock can't be ruled out and to be checked. In order to find whether or not the Petri Net is live; move along the markings of the Reachability tree. If any marking exists in the tree such that no transitions are enabled from that marking, then that marking represents a deadlocked state, and the Petri Net lacks the liveness property. Otherwise it is declared live. In our case there is no such deadlock situation appears in [p_1- p_9]. So we call our 2-phase commit protocol live.

However, problems arise when a reachability graph is used with loops in it. It may cause a particular place occupied with an infinite number of tokens. This would result in an infinite sized tree.

5. Conclusions

In this paper our focus is on building a robust distributed transaction handling protocol for efficient handling of distributed virtual data warehouse. Informational data from a relational database management system (RDBMS) or from some other data sources are called transactions and a kind of history generated out of transactions is stored in data warehouse (DW). When a Data Warehouse is connected with operational data base through the use of middleware is called as virtual Data Warehouse [14]. The virtual data ware house (VDW) will become the backbone of the distributed framework.

This paper presents a Petri Net model for the two phase commit protocol for transaction management in a distributed environment. There are inherent problems with the 2PC protocols like; blocking which reduces the availability of the resources. Ready state indicates participant waits for the coordinator and at this stage if the coordinator fails the site will be blocked until recovery. Again independent recovery is not possible; however, there exists recovery protocols for single site failures. The uncertainty factors like happening of an event, synchronization, resource sharing and communication are some of the most important aspects attempted to

formalize. We further propose to model the scalable and interoperable prototypes of such systems using high level net models.

7. References

[1] Bidyut Biman Sarkar and Nabendu Chaki, "Modeling and Analysis of Transaction Management for Distributed Database Environment using Petri Nets", Proc. of the 8th Int'l Conf. on Computer Information Systems and Industrial Management Applications (CISIM 2009), ISBN:978-1-4244-5612-3.

[2] Kaushal Chari, "Model composition in a distributed environment", Decision Support Systems archive, Elsevier Science, Volume 35(3), 2003, pp: 399-413, ISSN:0167-9236.

[3] Stefano Ceri and Giuseppe Pelagatti, "The management of distributed Transactions", Chapter 2, Distributed Databases-Principles & Systems, TMH, 1985, ISBN 0-07-066215-0

[4] B B Sarkar, N Chaki , "High level Net model for analyzing agent base distributed decision support system", Proc. of the IACSIT International Spring Conference, pp:339-346, 2009, ISBN 978-0-7695-3653-8.

[5] S. Kami Makki, and Ohio, "Distributed system:An Effective information Sharing Approach for Legacy Systems", JCIT journal volume 2 (2007), pp: 22-28 (3), ISSN : 1975-9320

[6] Ramchandani C., 1974, "Analysis of asynchronous concurrent systems by timed Petri nets", Technical report-120, (1974), Massachusetts Institute of Technology, Cambridge, ISBN 0-89791-070-2.

[7] W.M.P Vander Aalst, "Putting high-level Petri Nets to work in Industry", Elsevier Computers in Industry, Vol. 25, 1994, pp. 45-54., ISSN:0166-3615.

[8] Bernard Berthomieu and Miguel Menasche, "An Enumerative Approach for Analyzing Time Petri Nets", Proceedings IFIP, September 19-23 (1983), pp: 41-46, Elsevier, ISBN 0-444-86729-5.

[9] Bernard Berthomieu, Michel Diaz, "Modeling and Verification of Time Dependent Systems Using Time Petri Nets", IEEE Trans. on SE, Vol. 17(3), pp: 259 - 273, 1991 ISSN:0098-5589.

[10] Stefano Ceri and Giuseppe Pelagatti, "The management of distributed Transactions", chapter 7 of Distributed Databases, Principles & Systems, TMH, 1985, ISBN 0-07-066215-0

[11] Marian V, Iordache and Panos J. Antsaklis, "Supervisory Control of Concurrent Systems, Chapter 2, A Petri Net Structural Approach", Birkhauser Boston, ISBN-10 0-8176-4357-5.

[12] Tado Murata, "Petri Nets:Properties, Analysis and Applications", Proceedings of the IEEE, Volume 77, No .4, April 1989, pp:541-580, ISBN: 0-387-13723

[13] Barad M, "Timed Petri nets as a verification tool", Proc. of IEEE Winter Simulation Conf., pp: 547-554, 1998, ISBN: 0-7803-5133-9.

[14] Ammoura Ayman, Zaiane Osmar R., Yuan Ji, "Towards Framework for the virtual Data Warehouse", British National conference on databases, pp. 202-218, 2001, ISBN 3-540-42265-X.

[15] Venkata Padmavati Metta, Kamala Krithivasan and Deepak Garg, "Modeling Spiking Neural P systems using Timed Petri nets", Proceedings of the 8th International Conference on Computer Information Systems and Industrial Management Applications (CISIM 2009), ISBN:978-1-4244-5612-3.

8. Author Biography

Bidyut Biman Sarkar is a faculty member in MCA department of Techno India affiliated to West Bengal University of Technology. He has received his post graduate degree in Applied Mathematics from the University of Calcutta in the year 1978. Bidyut served IT industry in India, Singapore and other South Asian countries for more than 20 years. He is pursuing his Ph.D. studies in the areas of distributed computing. A good number of publications are already to his credit in national and international conferences and journals. He has also authored a few text books for the Engineering students at the under graduate level of Computer Science and Information Technology.

Nabendu Chaki is an Associate Professor in the Department of Computer Science & Engineering, University of Calcutta, Kolkata, India. He did his first graduation in Physics and then in Computer Science & Engineering, both from the University of Calcutta. He has completed Ph.D. in 2000 from Jadavpur University, India. Dr. Chaki has authored a couple of text books and more than 70 refereed research papers in Journals and International conferences. His areas of research interests include distributed systems, bio-informatics and software engineering. Dr. Chaki has also served as a Research Assistant Professor in the Ph.D. program in Software Engineering in U.S. Naval Postgraduate School, Monterey, CA. He is a visiting faculty member for many Universities including the University of Ca'Foscari, Venice, Italy. Dr. Chaki is a Knowledge Area Editor in Mathematical Foundation for the SWEBOK project of the IEEE Computer Society. Besides being in the editorial board for four International Journals, he has also served in the committees of more than 40 international conferences.