Provide a New Mapping for Deadlock Detection and Resolution Modeling of Distributed Database to Colored Petri Net

Masoomeh Ghodrati
Department of Computer Engineering, Science and Research Bushehr Branch, Islamic Azad University, Bushehr, Iran

Ali Harounabadi
Department of Computer Engineering, Tehran Center Branch, Islamic Azad University, Tehran, Iran

ABSTRACT
One of the most important applications of distributed systems is enabling resource sharing between systems. In such environments, if a sequence of procedures to control resource allocation is not possible to create a deadlock exists. Deadlock problem for a distributed database system that uses locking as a concurrency control algorithm, as there are inherent. The following new rule for the modeling of the proposed method using colored Petri nets is presented. In the model proposed the new rules for mapping TWFG with colored Petri nets for modeling the deadlocks detection and resolve. Colored Petri net is considered one of the most widely used formal methods capable of modeling a wide variety of distributed systems are concurrent. A lot of work being done to define the concurrency execution of transactions in Petri nets is that none of these methods of communication with how mapping TWFG with colored Petri nets for modeling the deadlocks detection and resolve.

General Terms
Databases, distributed, deadlocks, detection, colored Petri net, mapping.

Keywords
Resolution deadlock cycle, colored Petri net mapping, TWFG.

1. INTRODUCTION
In modern computer systems may be multiple transactions together to compete for a limited number of resources. Upon request, the request due to the unavailability of a resource if the resource cannot be assigned to a transaction, the transaction is expected to be the case. There may be circumstances, under which transactions are waiting, do not have a chance to change their situation. This situation can occur if the requested resource is being held by the transactions under the same expectation. This situation is called a deadlock.

In any database system which allows the simultaneous execution of transactions using locking protocols, deadlock can occur in the case of a distributed database system today. In a distributed database system, data will be accessed by concurrent transactions; these transactions to maintain consistency of the database are synchronized. This synchronization using concurrency control algorithms such as phase locking, timestamp ordering based, optimistic concurrency control or change in the fundamental algorithms are obtained.

Therefore, deadlock detection algorithms for centralized database systems are implemented and enforced.

Some algorithms for deadlock detection in centralized database systems are based on the discovery of implements cycles are a TWFG. Using a direct edge that determines the transactions is waiting in TWFG, if the cycle is detected, you can select a transaction and the sacrifice victim in the cycle, the cycle is broken. Transactions are usually allowed to start again with their original boxes. TWFG operation when a database is distributed across multiple sites is complicated. In a distributed database system, although a transaction may all activities at the site where the activity is started to execute, but may no longer original sites run. If this happens one agent at a remote site is created to show the transaction on its website, this agent is main part of the transaction concurrency control and recovery purposes.

Previous work carried out in the deadlock detection, the youngest of the transaction cycle is selected as a victim. But this choice has the following disadvantages:

• Transaction youngster who victims is selected, it may be important to the system. For example, the transaction has been sent from the management.

• Because of aborting Young transactions may be transaction that cause a deadlock in the system is always present and always create a deadlock.

• Transaction that has participated in more cycles and selected as victims may be has paramount importance for the system.

Ghodrati and Harounabadi in [24] provide a mechanism based on neighbor replication on grid for deadlock detection. After detecting a deadlock from proposed method, the deadlock victim's transaction is chosen optimally and aborting. The solution provided for selecting a victim to break the deadlock cycle in addition to the ID of priority importance for transaction systems is also considered. Younger transactions to avoid starvation constant factor each time aborting the transaction will be deducted from the amount of the transaction is restarted. The following new rules for the modeling of the proposed method using colored Petri nets are presented. In the model proposed the new rules for mapping TWFG to colored Petri nets modeling for the detection and resolve deadlock. In this paper used this algorithm.

The previous work done in this regard has been evaluated. After stating the disadvantages of previous methods, the proposed method is speech. The following new rules for the modeling of the proposed method using colored Petri nets are presented. Finally, after the conclusions, recommendations and limitations are described.

2. RELATED WORKS

Deadlock detection and elimination algorithms have been proposed so far are as follows.

2.1 Chandy Algorithm

The algorithm in [3] the TWFG for transactions at local sites and the probes used to detect global deadlock. If there is a deadlock, the probe is calculated by calling transaction Ti is executed. A probe is sent if a transaction is waiting for another transaction. Deadlock detection probe requests and responses are separate.

Each transaction at any stage eventually sends a probe. If probe starter achieve probe again, there is a deadlock. This design with deadlock detection is wrong, even if the transaction does not use 2-phase locking protocol.

2.2 Sinha Theory

This algorithm [4] is developed Chandy algorithm based on the priorities of transactions. Using the priorities, the number of messages required for deadlock detection considerably reduced.

The numbers of messages in the best and worst conditions are easily identified. This model includes transaction manager and data. The data manager is responsible for taking and releasing locks. A transaction requests a lock on a data item is sent to the manager if the application is not accepted, the administrator starts the deadlock is calculated. To do this a probe transaction is holding a lock on data item is sent as requested. If priority of holder transaction is higher than requester transaction then transaction inserts this probe to probe Q that holds this. When the probe is sent to the data manager data item, it waits. In this phase of the calculated deadlock, priority transactions to decide whether or not to release the probe will be used. If the Starter priority is greater than the item holder priority then will be release probe. When a transaction is waiting for a lock then all probe are expanded from its queue. when a data manager received self-starter probe again, dead lock detect. Since the probe includes younger transactions priority in cycle then younger transaction aborted.

2.3 Obermack Algorithm

This algorithm [5] makes and analyzes TWFG in each site directly. Distinct nodes at each site used a node that called external nodes. This node was used to display part of TWFG that is external (unknown) for site. Because at any time created waiting graph does not provide an overview of the Global TWFG. Detection algorithm in each site follows these steps:

• Build TWFG.

• To obtain and add the information received from other sites such as string to TWFG.

• Create edge waiting-for from external node to each node of representing agent from transaction that waiting for send on communication link.

• Create edge waiting-for from external node to each node of representing agent from transaction that waiting for received on communication link.

• Analysis of all cycle elements of TWFG.

• Select a victim to break each cycle that not containing external nodes. As a result any victim who is selected deleted all cycles that containing victim.

• For every cycle $E_x \rightarrow T_1 \rightarrow T_2 \rightarrow \dots T_x \rightarrow E_x$ contains nodes "outside", if the T1 transaction ID is greater than T_x then string Ex, T1, T2. . . Tx send to Tx site that are waiting to received.

2.4 Theory Menasce

This algorithm [6] was used initially to summarize TWFG, vertices represent transactions and arcs represent dependencies between transactions. The algorithm fails to detect some of the deadlock and possible to detect deadlock wrong.

The algorithm is described using the following rules:

Rule A:

Event: Transaction has T requests r_d source in S_k site and r_d kept by Transactions T1, T2. . . And T_n currently.

Action: an edge added from significant node to each transaction T1, T2. . . And Tn. if it causes a cycle in the TWF (k) then the deadlock exists.

Rule B:

Event: a pair of blocks (T, T ') has been received at site Sk.

Action: If the result was a cycle, deadlock is detected, then an edge from T to T ' in TWF (k) is added.

2.5 Ho Algorithm

In this algorithm [7], Table of transaction per sites hold information about resources held and waiting by local transactions and Selected a site as the central controller for deadlock detection for local transactions Waiting periodically. The problem with this approach is that it requires a number of 4n messages, where n is the number of sites in the system.

2.6 Kawazu Algorithm

This algorithm [8] is based on two-phase locking protocol. In first phase local deadlock and in second phase global in the absence of local deadlock are diagnosed.

This theory sees from ghost of deadlock harmful because every local waiting graph due to communication delays on time will not be collected. Also one transaction at a time, waiting is more than one source, after the global deadlock detection started even if no local deadlock is not detected, it may not detect some global deadlocks.

2.7 VGS Algorithm

This theory [9] is based on the lack of permission create such choices transaction deadlock victim is not the optimal way to resolve deadlock. This paper proposes a new algorithm for solving the deadlock that does not create any immediate repeal or rollback. The algorithm is based on "mutual sharing transactions". The disadvantage is that safe mode does not provide for concurrent execution of transactions.

2.8 Monjurul Algorithm

In this theory [1, 2], a distributed database system is a collection of data objects that are spread among a number of sites. These sites are connected with each other through messages. Each site includes a data object controller (scheduler) and data manager. This method is attempted transaction that is involved in more cycles to be selected as victims. The proposed technique is based on the following calculations:

• Linear Transaction Structure for each local site

• Distributed Transaction Structure for Global Resources connected transactions
• Priority ID for each transaction at each site
• Local and global cycles
• The victim aborted in the transaction cycle.

2.9 Ghodrati Algorithm

This algorithm [24] provides optimal choice of aborting the transaction as a deadlock victim. The proposed method is compared with previous work raises a good idea to choose a victim.

2.10 Other Algorithms

Bracha, Mitchell and Krivokapic algorithms for deadlock detection distributed database systems have been proposed that are not properly evaluated [1].

Some algorithms have been proposed in recent years [16-18] have previously described a new mechanism not only have issues. The algorithm presented in [14] and [23] has used the idea of the grid is not implemented. In [10] described a method for concurrency execution of transactions in colored Petri nets connection that no relation with TWFG mapping to Petri nets for detect and resolve the deadlock. As observed in previous work to resolve deadlock, the youngest of the transaction in cycle is selected as a victim. But this choice has disadvantages that in this paper provide a solution to solved this problem.

3. PROPOSED MODEL

A colored Petri net is widely used formal methods for the modeling capabilities in many types of distributed and concurrent systems. In the proposed method can be model and evaluate all the issues related to transactions resource requests and manage deadlock in databases distributed on colored Petri nets.

The proposed model for each transaction is a transaction manager intended to perform operations related to the transaction. Transaction manager hold information related to transaction, priority transaction ID and importance of transaction for system. Transaction manager assigned to a node. In the initial state, all transactions in TWFG create probe and send. These probes send for all transactions that current transaction waiting for them. Destination transaction after receive the probe send the probe for all transactions that are waiting for them.

RequestTrans indicates which transaction has received the probe. Each transaction manager after receiving probe will check whether or not the current transaction is same Transaction initiated probe. If not be probe starter then will examine whether the current transaction in the list (the list of transactions maintained by the current probe (ProbeQueue) exists or not. if current transaction exist in ProbeQueue then deadlock has been detected that starter transaction plays no role in deadlock. Thus, current transaction prevent of expect the probe to another transaction. Because possible declare a deadlock victim wrongly. It is noted that this deadlock will be detected by other probes. If current transaction exists in ProbeQueue then send probe will continue.

If starter transaction equal with current then deadlock cycle is detected. The victim transaction in probe that has been named as Victim Trans selected and aborted.

Each transaction in time to send probe, change probe values and send probe for all transactions that are waiting for current transaction. Current transaction before sending the probe, set own for last transaction defines in probe and adds to ProbeQueue. if Sign amount of current transaction is higher of the victim transaction then current transaction selected as a victim in the probe.

IN [24] for the selection of the victim transactions must be calculated for each transaction value of S.

$$S = \alpha \times Sign + (1 - \alpha) \times PTid \quad (2)$$

Sign: importance (significance) of the transaction to the system

PTid: Sequence number of transactions entry the system

α: important factor Select most important transaction as victim

PTid younger transaction number is higher. Transaction with Sign higher value is less important for systems. Transaction with Sign lower value is less important for systems.

Each transaction must be get priority of the transaction ID that called PTId and a significant amount of the transaction that called a Sign from transaction manager.

By entering a transaction in the system younger transaction may be a low priority; then this transaction whenever from system is aborting from transaction Sign constant β (multiplication factor of victim transactions) is low.

$$Sign = Sign-\beta \quad (3)$$

This causes the victim to begin again more significant transactions and transactions to prevent starvation.

It should be noted that the appropriate values of β and α are determined by the distributed database management. Determine appropriate values for β and α has a significant impact on system resource usage by transactions. For example, if a low priority transaction that is older and has a lot of resources of the system used to be the victim, the system will impose large costs.

As a result, the values of β and α must be determined in such a way that the victimization of young and old transactions and transactions are less important to create more balance.

This article proposed a new method for mapping model based on neighbor replication on grid to colored Petri net to modeling identify and resolve the deadlock. Since the model is based on TWFG can be done by mapping TWFG to colored Petri net is based on the Grid. Fig.4 Proposed models for mapping TWFG with colored Petri net is presented to modeling identify and resolve the deadlock.

In Fig.4, status of implementation of the transaction mapped to Execute place and victim transaction mapped to victim place. Type of color set selected for these places is SITExTRANSACTIONxTIdxTSign. AbortVictim transition is responsible for the aborted victim's transaction. This transition will remove the victim from the color set TWFG place. ExitTransition transition used to end transaction after it has been run. This transition removed transaction color from color set TWFG place. SequenceOfRequests Place task is responsible for the transaction request signals. The location of the color of the color set are selected PROBE.

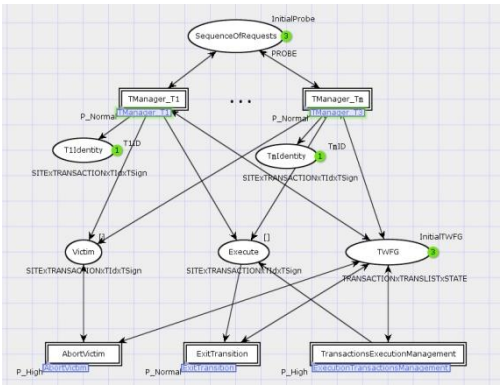

Fig.4: Proposed model for detecting and resolve deadlock

Color sets for this model are as follows:

```
val TransactionsNO = 3;
val SitesNO = 3;
colset  STATE= string;
colset TRANSACTION = index Trans with 1. . TransactionsNO;
colset SITE = index Site with 1. . SitesNO;
colset SiteLIST = list SITE;
colset SITExTRANSACTION = product SITE*TRANSACTION;
colset TRANSLIST = list SITExTRANSACTION;
colset SITExTRANSACTIONxSITExTRANSACTION=product
SITExTRANSACTION*SITExTRANSACTION;
colset TRANSACTIONxTRANSLISTxSTATE= product
SITExTRANSACTION*TRANSLIST*STATE;
colset TWFGList = list
SITExTRANSACTIONxSITExTRANSACTION;
colset TId= int;
colset TSign= real;
colset SITExTRANSACTIONxTIdxTSign= product
SITExTRANSACTION*TId*TSign;
colset PROBE = product
SITExTRANSACTION*SITExTRANSACTION
*SITExTRANSACTION *TRANSLIST*
SITExTRANSACTIONxTIdxTSign;
colset  Boolean =bool;
colset  SEQUENCE = int;
```

Variables for greater flexibility in system modeling are introduced. Variables are defined for colors set required to implement the model variables are defined as follows:

```
var  T, Tv, NT : SITExTRANSACTION;
var Victim, NVictim, ReadyTrans :
SITExTRANSACTIONxTIdxTSign;
var  state, Nstate : STATE;
var TL, NTL, ProbeQueue, NProbeQueue : TRANSLIST;
var S, R, StarterT, LastT, RequestT: SITExTRANSACTION;
var Otwfg, Ntwfg, twfg: TRANSACTIONxTRANSLISTxSTATE;
var Ids, Idv: TId;
var Ss, Sv : TSign;
var Abort, BroadCast, Ready : Boolean;
```

Transition capacity led to the assignment of variables to colors from input edge markup.
Calling functions in the model are defined as follows:

```
fun intToReal i = (IntInfToReal FloatPoint (IntInf. fromInt i))
```

IntToReal function converts the input value from int to real. The number of decimal digits displayed is determined by FloatPoint. Here FloatPoint out of 5.

```
fun getTransIndex( t, h::L ) : int =
 let   val i =0
 in    if ( h=t) then   i
       else
            if  (getTransIndex(t, L) <> ~1) then
                 getTransIndex(t, L) +1
            else    ~1
 end
| getTransIndex( _, [] ) = ~1;
```

getTransIndex function return position index of T transaction in the list h :: L.

```
fun eliminateTrans (T, L)=   let   val  index = getTransIndex(T, L)
   in   if (index <> ~1)  then                 List. take(L, index)^^List.
drop(L, index+1)
         else   L      end
```

EliminateTrans removed T transaction from the list L.

```
fun isExists(T, TL ) =    let  val  n = getTransIndex(T, TL)
   in   if n <> ~1 then      true      else            false    end ;
```

IsExists function will check whether existing T transaction in the TL transactions list is.

```
fun BroadCastMessage(translist, starterT, s, probequeue, victim = (
if translist=[] then
      empty
else      1`(starterT,      s,      hd(translist),      probequeue,      victim)
++BroadCastMessage(tl(translist), starterT, s, probequeue, victim );
```

BroadCastMessage function creates probes.

```
fun  CheckVictimIsInRequestListTransaction  ((Vsite,  Vtrans),  (t,
translist, state= ((
let  in if isExists((Vsite, Vtrans), translist ) then
     true else       false end;
```

CheckVictimIsInRequestListTransaction function will check availability victim transition in request transitions in TWFG place. If available, the function returns true and false otherwise.

```
fun CheckVictim ((Ssite, Strans), probequeue, ((Vsite, Vtrans), Vid,
Vsign) : SITExTRANSACTIONxTIdxTSign,
((Csite, Ctrans), Cid, Csign) : SITExTRANSACTIONxTIdxTSign,
transList)=
val a=0. 5; idv=intToReal Vid; idc=intToReal Cid; sV = a * Vsign + (
1. 0 - a ) * idv; sC = a * Csign + ( 1. 0 - a ) * idc;  index =
getTransIndex((Csite, Ctrans), probequeue)
in
if ((Ssite, Strans)= (Csite, Ctrans) andalso probequeue <> []) then
   if (sV>sC) then
        (((Vsite, Vtrans), Vid, Vsign), ((Csite, Ctrans), Cid, Csign),
probequeue, true, false, false)
   else
        (((Csite, Ctrans), Cid, Csign), ((Csite, Ctrans), Cid, Csign),
probequeue, true, false, false)
else
   if (transList <> [] ) then
      if (index = ~1)  then
       if (sV>sC) then
            (((Vsite, Vtrans), Vid, Vsign), ((Csite, Ctrans), Cid,
Csign), (Csite, Ctrans) :: probequeue, false, true, false)
        else
            (((Csite, Ctrans), Cid, Csign), ((Csite, Ctrans), Cid, Csign),
(Csite, Ctrans) :: probequeue, false, true, false)
      else
  (((Vsite, Vtrans), Vid, Vsign), ((Csite, Ctrans), Cid, Csign), (Csite,
Ctrans) :: probequeue, false, false, false)
   else
       (((Vsite, Vtrans), Vid, Vsign), ((Csite, Ctrans), Cid, Csign),
(Csite, Ctrans) :: probequeue, false, false, true)
end;
```

The model in Fig.5 implements a new method for mapping TWFG to colored Petri net is deadlock detection and resolution; TWFG requests for third transaction has implemented.

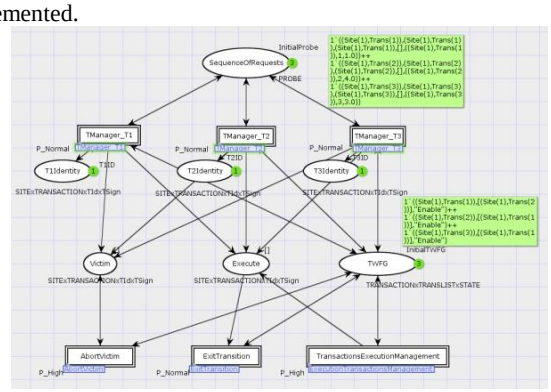

Fig.5: The model has been implemented using CPN

In Fig.6, T1 transaction mapped to CPN.

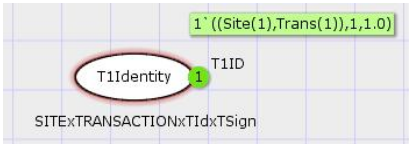

Fig.6: Mapping T1 transaction to CPN

In Fig.7, each transaction manager mapped to CPN.

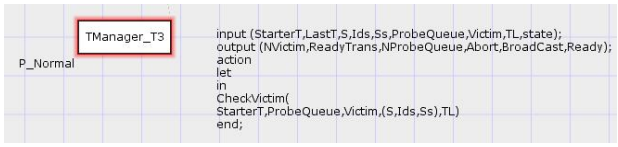

Fig.7: Mapping each transaction manager to CPN

In Fig.8, wait-for graph mapped to CPN.

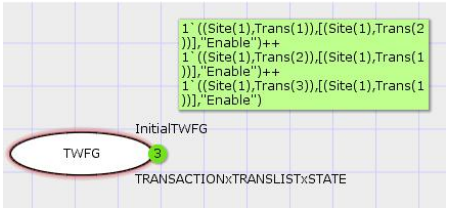

Fig.8: Mapping wait-for graph to CPN

3.1 Mapping T1 transaction manager

CPN model module T1 transaction is a transition that example is as follows.

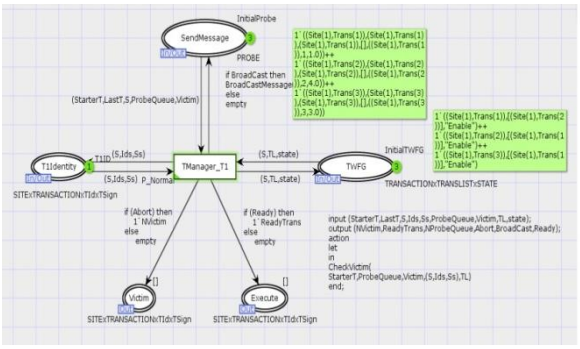

Fig.9: CPN model T1 transaction manager

3.2 Mapping ExitTransition module

CPN model ExitTransition module is as follows.

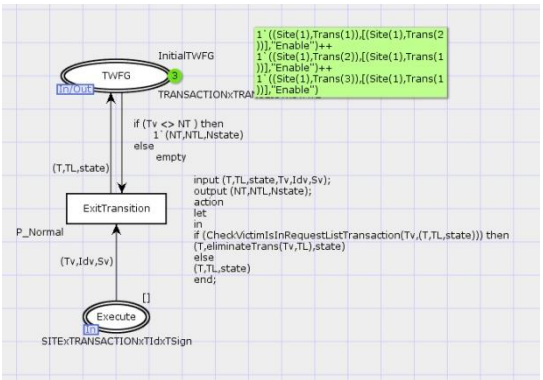

Fig.10: CPN Model module ExitTransition

3.3 Mapping module ExecutionTransactionsManagement

CPN model ExecutionTransactionsManagement module is as follows.

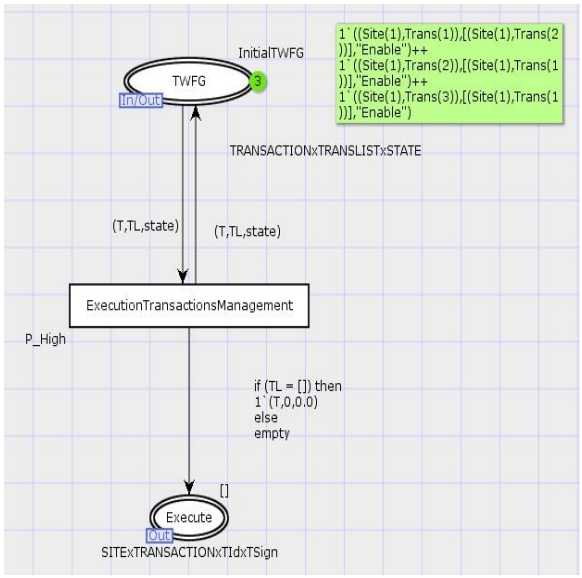

Fig.11 : CPN Model module ExecutionTransactionsManagement

3.4 Mapping module AbortVictim

CPN model AbortVictim module is as follows.

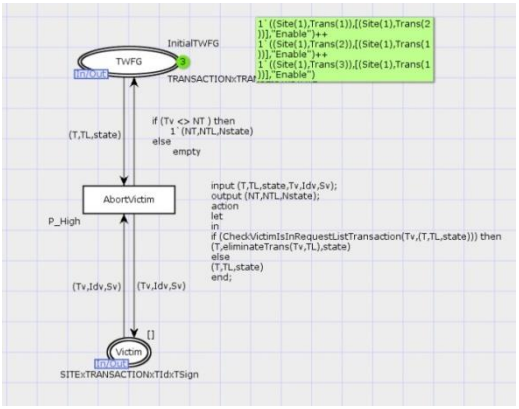

Fig.12: CPN model module AbortVictim

3.5 Implementation model places colors set

CPN is a set of colors assigned to each specifies place that can be included. Mark a place as a "multi- set" that the presented model is defined as follows:

val P_High = 1000;
val P_Normal = 100;
val T1Requests = 1 `((Site (1), Trans (1)), [(Site (1), Trans (2))], " Enable "; (
val T2Requests = 1 `((Site (1), Trans (2)), [(Site (1), Trans (1))], " Enable "; (
val T3Requests = 1 `((Site (1), Trans (3)), [(Site (1), Trans (1))], " Enable "; (
val InitialTWFG = T1Requests ^ ^ T2Requests ^ ^ T3Requests;
val T1ID = 1 `((Site (1), Trans (1)), 1, 1. 0 (;
val T2ID = 1 `((Site (1), Trans (2)), 2, 4. 0 (;
val T3ID = 1 `((Site (1), Trans (3)), 3, 3. 0 (;
val InitialSignificance = T1ID ^ ^ T2ID ^ ^ T3ID;
val FloatPoint = 5;
val InitialProbe =

1 `)) Site (1), Trans (1)), (Site (1), Trans (1)), (Site (1), Trans (1)), []: TRANSLIST, ((Site (1), Trans (1)), 1, 1. 0)) + +1 `)) Site (1), Trans (2)), (Site (1), Trans (2)), (Site (1), Trans (2)), []: TRANSLIST, ((Site (1), Trans (2)), 2, 4. 0 ((
+ +1 `)) Site (1), Trans (3)), (Site (1), Trans (3)), (Site (1), Trans (3)), []: TRANSLIST, ((Site (1), Trans (3)), 3, 3. 0))

Proof of correctness:
Deadlock detection and resolution algorithms must make the following two conditions have to be considered:
• Security - The algorithm correctly diagnoses and solves the deadlock.
• Life - or dissolve detection algorithm identified all the deadlock ends in the limited time to solve.
This condition is usually examined under the assumption that all abortions there in the system created by the algorithm, wherever there is no spontaneous abortion. It is proved that any algorithm has the ability to play up the security conditions are not considered unless spontaneous abortion [21].
The model of detection and resolution of deadlock is working correctly.

4. COMPARE THE PROPOSED METHOD WITH EXISTING METHODS

Obermack algorithm does not work correctly because it detect wrong deadlock because general overview of TWFG global does not display any moment. Menasce theory fails to recognize some cycles may cover deadlock wrong. HO algorithm transaction tables and table references per site to keep information resource uses. The disadvantage to this approach is that it requires the 4n messages, where n is the number of sites in the system. Kawazu the ghost deadlock graph algorithms suffer because waiting times due to communication delay time is not collected and not found a local deadlock, some of the diagnosed may not be deadlock ends Global. Transaction deadlock detection algorithms presented in most of the young people who choose to cycle as a victim. Younger transactions may be important if the victim is selected, the system is high. It may also be because young aborting transaction, the transaction is responsible for the deadlock, the deadlock created in the system and constantly present and it is possible that the transaction more cycles, and the company has been selected victim, the system of is high.
In algorithm used in the model that has been presented by Ghodrati, the iterative technique is used neighbor replication on grid. This method for solving the problems should be a priority identifier for each transaction in addition, the amount of the transaction to the system (Sign) will also be considered. Sign aborting the transaction whenever the system is its low transaction constant β. To choose a victim, the system must strike a balance between ID and Sign establish priorities. This technique ensures that deadlock detection and global deadlock nothing to do with the local algorithm does not detect any false deadlock actual deadlock is detected.
A lot of work being done to define the concurrency execution of transactions in Petri nets is that none of these methods of communication with Petri nets for modeling how to detect and resolve the mapping TWFG not deadlock. This paper presents an approach to transforming TWFG the Petri net. The probe has been used to implement some of the previous methods. The victim is chosen to account for the transaction in the event of deadlock detection is no need to re- route the transaction to select the victim is dead.

5. CONCLUSION

In this paper, a new method for mapping TWFG to colored Petri net for modeling detect and resolve deadlock using colored Petri net is presented better to use.
Petri nets for modeling and analyzing systems that have issues with parallelism, and synchronization face encounters are very powerful and convenient. Knowing the mapping rules presented TWFG can easily be converted to a colored Petri nets. Mapping rules are applied in a real case shows the feasibility of mapping rules.

6. LIMITATIONS

In this paper, the bases of all decisions based on the current state of the system and are TWFG. Given that no comprehensive assessment to compare all available methods to detect and resolve deadlock in distributed databases with the same condition does not exist, no comprehensive comparison of all the methods of different states of.

7. SUGGEST FUTURE WORK

As future work, we can consider the following:
• The use of intelligent algorithms to determine the optimal value of α (the coefficient is chosen as the victim of the most important transactions) and β (a factor of increasing importance victim restarting transaction) given the current state of the database.
• Mapping of all operations related to distributed databases to colored Petri nets.

8. REFERENCES

[1] B. M. Monjurul Alom, Frans Alexander Henskens and Michael Richard Hannaford, "Optimization of Detected Deadlock Views of Distributed Database", First International Conference on Data Storage and Data Engineering, Vol. 41, pp. 44-48 2010.

[2] B. M. M. Alom, F. Henskens, and M. Hannaford, "Deadlock Detection Views of istributed Database", in 6th International conference on Information Technology & New Generartion (ITNG-2009) Las Vegas, USA. IEEE Computer Society, pp. 730-737, 2009.

[3] K. M. Chandy, J. Misra, and L. M. Hass, "Distributed Deadlock Detection", ACM Transaction on Computer Systems, Vol. 1, Issues. 2, pp. 144-156, 1983.

[4] M. K. Sinha and N. Natarjan, "A Priority Based Distributed Deadlock Detection Algorithm" IEEE Transactions on Software Engineering, Vol. 11, Issues. 1, pp. 67-80, 1985.

[5] R. Obermarck, "Distributed Deadlock Detection Algorithm", ACM Transaction on Database Systems, Vol. 7, Issues. 2, pp. 187-208, 1982.

[6] D. A. Menasce and R. R. Muntz, "Locking and Deadlock Detection in Distributed Data Bases" IEEE Transactions on Software Engineering, Vol. 5, Issues. 3, pp. 195-202, 1979.

[7] G. S. HO and C. V. RAMAMOORTHY, "Protocols for Deadlock Detection in Distributed Database Systems " IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, Vol. 8, Issues. 6, pp. 554-557, 1982.

[8] S. Kawazu, S. Minami, K. Itoh, and K. Teranaka, "Two-Phase Deadlock Detection Algorithm in Distributed Databases", VLDB, the fifth International Conference on Very Large Data Bases – Vol. 5, pp. 360-367, 1979.

[9] Kunwar Singh Vaisla and Menka Goswami and Ajit Singh, "VGS Algorithm - an Efficient Deadlock Resolution Method", International Journal of Computer Applications, Vol. 44, Number. 1, pp. 29-33, April 2012.

[10] Saeid Pashazadeh, "Modeling and Verification of Deadlock Potentials of a Concurrency Control Mechanism in Distributed Databases Using Hierarchical Colored Petri Net ", International Journal of Information and Education Technology, Vol. 2, No. 2, pp. 77-82, April 2012.

[11] B. B. Sarkar and N. Chaki, " Transaction Management for Distributed Database using Petri Nets", International Journal of Computer Information Systems and Industrial Management Applications (IJCISIM), ISSN. 2150-7988, Vol. 2, pp. 069-076, 2010.

[12] K. Maabreh and and A. Hamami, "Implementing New Approach for Enhancing Performance and Throughput in A Distributed Database", Vol. 10, Issue. 3, pp290-296, IAJIT, 2011.

[13] D. Gupta and V. K. Gupta, " Approaches for Deadlock Detection and Deadlock Prevention for Distributed systems", Research Journal of Recent Sciences, Vol. 1, pp. 422-425, 2012.

[14] Noriyani Mohd Zin, A. Noraziah, A. H. Beg, Ainul Azila Che Fauzi, "Deadlock Detection and Resolution in Neighbour Replication on Grid", IACSIT, Vol. 5 No. 22, pp. 350-357, 2011.

[15] V. Geetha and N. Sreenath, " Distributed Deadlock Detection using Fault Informing Probes ", International Journal of Computer Applications, Vol. 41, No. 8, pp. 6-11, March 2012.

[16] Arun Kumar Yadav and Dr. Ajay Agarwal, "A Distributed Architecture for Transactions Synchronization in Distributed Database Systems ", International Journal on Computer Science and Engineering, Vol. 02, No. 06, p. 1984, 2010.

[17] Himanshi Grover and Suresh Kumar, "ANALYSIS OF DEADLOCK DETECTION AND RESOLUTION TECHNIQUES IN DISTRIBUTED DATABASE ENVIRONMENT", International Journal of Computer Engineering & Science, Vol. 2, Issue 1, pp. 17-25, Sept 2012.

[18] Srinivasan Selvaraj and Rajaram Ramasamy, "An Efficient Detection and Resolution of Generalized Deadlocks in Distributed Systems", International Journal of Computer Applications, Vol. 1 – No. 19, pp. 1-7, 2010.

[19] M. Tamer Ozsu and Patrick Valduries, "Principles OF Distributed Database System", Springer, Third Edition, pp. 387-394, 2011.

[20] Pooja Sapra and Suresh Kumar and R K Rathy, "Deadlock Detection and Recovery in Distributed Databases", International Journal of Computer, Vol. 73, No. 1, pp. 32-36, July 2013.

[21] Gupta and Swati, "Deadlock Detection Techniques in Distributed Database System", International Journal of Computer Applications. Jul2013, Vol. 74 Issue 1-21, pp41-45, 2013.

[22] Knapp. E., "Deadlock Detection in Distributed Databases", ACM Computing Surveys, Vol. 19, no. 4, pp. 303-328, Dec. 1987.

[23] Noriyani Mohd Zin, A. Noraziah, Ahmed N. Abdalla and Ainul Azila CheFauzi, "Solving Two Deadlock Cycles through Neighbor Replication on Grid Deadlock Detection Model ", Journal of Computer Science, Vol. 8, No. 2, pp. 265-271, 2012.

[24] Masoomeh Ghodrati and Ali Harounabadi, "A New Method for Optimization of Deadlock resolution of Distributed Database with Formal Model", International Journal of Electronics Communication and Computer Engineering, Vol. 5, no. 1, pp. 220-228, Jun. 2014.