Modeling Distributed Active Database Using CTPN

BULBUL MUKHERJEE
Dept. of Computer Science and Engineering, Bengal Institute of Technology, Kolkata, West Bengal, India,
mebulbulmukherjee@gmail.com

RANJAN DASGUPTA
Dept. of Computer Science and Engineering, National Institute of Technical Teachers’ Training and Research, Kolkata,
West Bengal, India
ranjandasgupta@ieee.org

Abstract: - This paper deals with color time Petri Net to model the Distributed Active Database. Here we discuss the complex operation sequences of the distributed active database including ‘indefinite waiting period’ of some responses and model it by using Color Time Petri Net (CTPN). Benefits of such modeling have also been mentioned.

Key-Words: - Petri Net, Color Time Petri Net, Distributed Active Database, ECA Rule.

1 Introduction
Petri Net[1][2][3] are one of the very useful graphical and mathematical representation and are being used to model complex and concurrent systems. Several variations of Petri Net had already been proposed which can handle more and more complex problems. These modifications include introduction of color(data), time etc. in the Petri Net. In this paper we discuss ‘Distributed Active Database’ problem and model it with the help of Color Time Petri Net (CTPN)[4]. The CTPN is very much applicable over here as the basic operations of distributed active database deal with various data elements(value) and based on these values, transitions are taking place. In case of some transitions, waiting time also plays a critical role and hence time color Petri Net becomes a natural choice.

2 Petri Net
Concurrency can pose a problem when many entities (people, machines, processing threads) use (share) the same resource (or a limited number of resources). A Petri net, introduced by C. A. Petri in 1962 is formally defined as a 5-tuple \( N = (P, T, I, O, M_0) \), where

1. \( P = \{p_1, p_2, \ldots, p_m\} \) is a finite set of places;
2. \( T = \{t_1, t_2, \ldots, t_n\} \) is a finite set of transitions, \( P \cup T \neq \emptyset \), and \( P \cap T = \emptyset \);
3. \( I: P \times T \to N \) is an input function that defines directed arcs from places to transitions, where \( N \) is a set of nonnegative integers;
4. \( O: T \times P \to N \) is an output function that defines directed arcs from transitions to places; and
5. \( M_0: P \to N \) is the initial marking.

A marking in a Petri net is an assignment of tokens to the places of a Petri net. Tokens reside in the places of a Petri net. The number and position of tokens may change during the execution of a Petri net. The tokens are used to define the execution of a Petri net. The type of Petri nets described up to this point is called Place/Transition nets. They have a number of limitations:

a. Inability to model similar (but not identical) processes using one net
b. All tokens are identical
c. No way to represent additional properties – there is no way to associate any additional data with a token.

In order to overcome these problems, a number of solutions extending the initial approach i.e high level Petri Nets have been proposed. Colored Petri Nets(CPN)[4], introduced by Kurt Jensen in (Jensen 1981) is a discrete-event modeling language combining Petri nets with the functional programming language Standard ML[4]. The state of the modeled system is represented by the places. Each place can be marked with one or more tokens, and each token has a data value attached to it. This data value is called the token color. It is the number of tokens and the token colors on the individual places which together represent the state of the system. This is called a marking of the CPN model, while the tokens on a specific place constitute the marking of that place. By convention, we write the names of the places inside the ellipses. The set of possible token colors is specified by means of a type (as known from programming languages), and it is called the color set of the place.

By firing a transition, tokens are removed from the input places and added to the output places in the same way as that in original Petri nets, except that a functional dependency is specified between the color of the transition firing and the colors of the involved tokens. The color attached to a token may be changed by a transition firing and it often represents a complex data-value.
The Color Time Petri Net (CTPN) or timed CPN extends the framework of the original PN by adding color and time attributes to the net. The time attribute allows various time-based performance measures to be conducted in the system model. A time-delay can be assigned to either places or transitions to model the time elements in a system.

The main difference between timed and un-timed CPN models is that the tokens in a timed CPN model—in addition to the token color—can carry a second value called a time stamp. This means that the marking of a place where the tokens carry a time stamp is now a timed multi-set specifying the elements in the multi-set together with their number of appearances and their time stamps. Furthermore, the CPN model has a global clock representing model time. The distribution of tokens on the places together with their time stamps and the value of the global clock is called a timed marking. In general, a time stamp can be a non-negative integer or real. In the current implementation of CPN Tools, only non-negative integers are supported. The time stamp tells us the time at which the token is ready to be used, i.e., the time at which it can be removed from the place by an occurring transition. The tokens on a place will carry a time stamp if the color set of the place is timed. A color set is declared to be timed using the CPN ML keyword timed[3].

3 Active Database

Active databases[5] are able to monitor and react to specific circumstances according to its relevancy to an application. An active database system must provide a knowledge model (i.e., a description mechanism) and an execution model (i.e., a runtime strategy) for supporting this reactive behavior. A common approach for the knowledge model uses rules that have up to three components: an event, a condition, and an action[5]. The event part of a rule describes a happening to which the rule may be able to respond. The condition part of the rule examines the context in which the event has taken place. The action describes the task to be carried out by the rule if the relevant event has taken place and the condition has evaluated to true.

Briefly, events can be classified into: i) primitive events and ii) composite events. Primitive events refer to elementary occurrences which are pre-defined in the system. Primitive events can be further decomposed into database events, time events, transaction events, method events etc. A composite event is a set of primitive events or composite events related by defined event operators.

3.1 Events in Distributed Active Database

Four types of events can be identified in a distributed database system[6]. Local Primitive Events are events that are predefined in that application using primitive event expressions and can be detected by a mechanism embedded in the system. Local composite events are composed of local primitive events and other local composite events by applying event operators. Global primitive events are events that are defined and detected outside of the current application but are referenced/used by the current application in a distributed environment. Global composite events are related to event occurrences from many sites (including the local site). They are constructed with local primitive events, local composite events, global primitive events and other global composite events.

4 Related Work

Active database capability most frequently uses triggers that execute actions based on the Event-Condition-Action (ECA) Model. Several event algebras have been developed, e.g. Snoop[7], SAMOS[8], ODE[9]. SAMOS combines active and object-oriented features in a single framework using colored Petri nets. Snoop is an event specification language which defines different restriction policies that can be applied to the operators of the algebra. The Global Event Detector (GED) (Chakravarthy and Liao, 2001) detects events in a distributed environment through a client/server architecture that minimizes message communication and allows for event registration and notification. GED extends the active capabilities of the Sentinel[10] active OODBMS.

In SAMOS, a very complex structure is used to represent a sequence of composite event. This will result in an extremely huge CPN model, which will be inconvenient for large rule-base development, and will be difficult to be implemented and managed. A software platform has been developed, which can generate a Conditional Color Petri Net (CCPN) model[11] automatically from a text file of ECA rule description, and communicate with a traditional database system when an event is detected from the database or an action command is generated by the CCPN simulator. This CCPN model will overcome the disadvantage of using a redundant structure to specify a temporal relation between primitive events since this information can be considered as a condition on transitions.

Baba-hamed.L and Belbachir.H (2005)[12] propose a method of termination analysis of active rules based on Petri Nets (PN) called as Extended Coloured Petri Net (ECPN) and give an object oriented
representation to implement it. Latifa Baba-Hamed (2008) has done a comparative study of the above method with the most known methods available in the literature for detecting non-termination. Li and Marin presented an approach based on coloured Petrinets named CCPN for modeling the active database behaviour. Incidence matrix of PN theory is used to find cyclic paths existing in CCPN[13]. Cycles which satisfy some theorems given by the authors are deleted. If there is no cycle in the CCPN, the termination of the corresponding set of rules is guaranteed. Nevertheless, this approach did not consider the priority of rules. An enhanced ECA rule, called ECA-AA is defined as a four tuple \((Event, Condition, Action, Alternative Action)\) as ECA Mechanism for Distributed Active Database Systems with High Autonomy Degree is done in the research paper[14]. The ultimate goal of this enhancement is a more flexible ECA mechanism, which allows us to continue work even if subsystems are not reachable.

5 Scope of the Work

Conventional database management systems are passive. Data are created, retrieved, modified, and deleted only in response to operations issued by users. By contrast, context-aware computing[ 15] is required to automatically carry out some services in response to certain changes in the real world, once conditions are being satisfied. Therefore, it needs some facets that active database systems have. The basic idea behind another emerging concept Ambient Intelligence (AmI)[16] is that by enriching an environment with technology (e.g., sensors and devices interconnected through a network), a system can be built such that acts as an "electronic butler", which senses features of the users and their environment, then reasons about the accumulated data, and finally selects actions to take that will benefit the users in the environment. Ambient intelligence (AmI) is a new research area builds upon advances in sensors and sensor networks, pervasive computing, and artificial intelligence and is for distributed, non-intrusive, and intelligent software systems. Modern large-scale applications, such as e-commerce, Internet or Intranet applications, enterprise application integration (EAI), and emerging pervasive systems, can effectively benefit from an active mechanism but conventional active mechanisms have been designed for centralized systems, so it must be adapted to meet the requirements imposed by this generation of large-scale heterogeneous applications. Thus modeling the overall functionality of a distributed active database is very much significant.

Pervasive computing[17][18][19] is characterized by a high degree of heterogeneity, mobility and support of distributed computing architectures which works on online connections. However, remote systems may be inaccessible occasionally. In case of active database systems this often leads to undecidable ECA conditions and rules. In distributed active databases, both event evaluation and condition evaluation can have an indefinite result because of unavailable subsystems. This leads to an abort after a timeout even if this is not desirable or necessary at all. By enhancing ECA rules with additional actions for the case of undecidability of ECA conditions, it becomes possible for active databases to react alternatively, which makes the entire ECA mechanism more robust. Our proposed model by CTPN can help to design and implement the distributed active database functionality which is very much relevant in today’s emerging technologies of AmI.

6 Proposed CTPN based Model of Distributed Active Database

We propose the model of the distributed active database which has the client server architecture supported with decentralized event detection and decentralized rule base.

- Under the client more than one application can run at a time.
- There will be provision of color set to identify a specific application, a specific client, event identification and the type of event (whether it is local or global and primitive or composite).
- The server can send event notification to clients by detecting any external event also which happening outside the database itself.
- Each application needs to reveal what events it is interested in from other applications. Then that application of that client can send requests to the server, the server can send response if the requested event is already detected, otherwise it can send requirements to the local service, process it and sends the necessary information back to the client.
- After completion of the event detection the rule processing is done by the client to trigger a specific rule and the condition evaluation process is occurred.
- Similar to the events, a condition may also be composed of different sub conditions which may be evaluated either in local site or globally through the server.
- Due to some reasons like System site failures or Network failure or Low network bandwidth etc. distributed event detection or condition evaluation can fail in one remote system and for
that the total procedure will be affected or delayed.

After completion of each event detection or action the application of the client will send message to the server.

6.1 Temporal Constraints:
In this model a deadline is there for each event detection process i.e maximum allowable time to detect an event and maximum allowable time to evaluate one condition.

Fig 1 CTPN model of distributed active database

We assume,

TE represents the predefined maximum time length for completion of event detection of a composite event.

If (1) does not satisfy the event detection process as a whole will go for an indefinite state. If (2) does not satisfy the condition evaluation process as a whole will go for an indefinite state.
Table 1: Places, Transitions, Tokens and their Purposes

<table>
<thead>
<tr>
<th>Places</th>
<th>Transitions</th>
<th>Purpose of transition</th>
<th>Tokens Passed</th>
<th>Purpose of token/guard conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1 SERVER</td>
<td>T1</td>
<td>When any external event occurs the server has to notify it to the client or server interacts with the client in response of client’s event detection request. Or condition evaluation request.</td>
<td>Ex, Ev (boolean), Ex, Dt_Req (boolean), Con, Ev_Req (boolean)</td>
<td>Ex, Ev = TRUE (happening of an external event) Ev, Dt_Req=TRUE (server responses on client’s event detection request) Con, Ev_Req=TRUE (server responses on client’s condition evaluation request)</td>
</tr>
<tr>
<td>P2 CLIENT INTERFACE</td>
<td>T2</td>
<td>The client interface can reply to the server in response of any event detection request or condition evaluation request placed by server.</td>
<td>Ev, Type, Dis (string) Ev, Dt, Status[i] (integer array)</td>
<td>[Ev, Type, Dis = GLOBAL AND Ev, Dt, Status[i] = 1] OR [Cond, Type, Dis = GLOBAL AND Con, Ev, Status[i] = 1] (client responses on server’s event detection request or condition evaluation request)</td>
</tr>
<tr>
<td>P1 SERVER</td>
<td>T3</td>
<td>Request for global event detection may come from the client.</td>
<td>Ev, Type, Dis (string) Ev, Dt, Status[i] (integer array)</td>
<td>[Ev, Type, Dis = GLOBAL] AND [Ev, Dt, Status[i] = 1] (client sends request to server for global event detection)</td>
</tr>
<tr>
<td>P2 CLIENT INTERFACE</td>
<td>T4</td>
<td>Each event detection completed and each event detection result is true go so for rule firing</td>
<td>Ev, Type, Dis (string) Ev, Dt, Status[i] (integer array)</td>
<td>[Ev, Type, Dis = LOCAL OR Ev, Type, Dis = GLOBAL] AND [Ev, Dt, Status[i] = 1 for i=1 to n] AND [Event, Dt, Res[i] = 1 for i=1 to n] (all local and global events detected and each event detection result is true)</td>
</tr>
<tr>
<td>P2 CLIENT INTERFACE</td>
<td>T5</td>
<td>Client interface will wait for each sub event detection for maximum time allotted for that event detection.</td>
<td>Ev, Dt, MaxTime (integer)</td>
<td>Ev, Dt, MaxTime is the deadline for detecting an event. All tokens will be passed from P2 to P3 when Ev, Dt, MaxTime is over.</td>
</tr>
<tr>
<td>P4 EVENT DETECTION TRUE AND RULE FIRED P6 ACTION</td>
<td>T6</td>
<td>After evaluating the conditions if all evaluated as true then go for action.</td>
<td>Cond, Type, Dis [string] Con, Ev, Status[i] (integer array)</td>
<td>[Cond, Type, Dis = LOCAL OR Cond, Type, Dis = GLOBAL] AND Con, Ev, Status[i] = 1 for i=1 to n AND Con, Ev, Res[i] = 1 for i=1 to n (all local and global sub conditions evaluated and each condition evaluation result is true)</td>
</tr>
<tr>
<td>P4 EVENT DETECTION TRUE AND RULE FIRED</td>
<td>T7</td>
<td>Request to server for global condition evaluation.</td>
<td>Cond, Type, Dis [string] Con, Ev, Status[i] (integer array)</td>
<td>Cond, Type, Dis = GLOBAL AND Con, Ev, Status[i] = FALSE some global conditions not evaluated so client sends request to server for that.</td>
</tr>
<tr>
<td>P4 EVENT DETECTION TRUE AND RULE FIRED</td>
<td>T8</td>
<td>Client interface will wait for each sub condition evaluation process for maximum time allotted for condition evaluation.</td>
<td>Cond, MaxTime (integer)</td>
<td>Cond, MaxTime is the deadline for evaluating a condition. All tokens will be passed from P4 to P5 when Cond, MaxTime is over.</td>
</tr>
<tr>
<td>P3 EVENT DETECTION FALSE OR INDEFINITE P8 NO ACTION</td>
<td>T9</td>
<td>All events detections are definite but result of some events are false So go for no action.</td>
<td>Ev, Type, Dis [string] Ev, Type, Dis [string] Ev, Dt, Status[i] (integer array)</td>
<td>[Ev, Type, Dis = LOCAL OR Ev, Type, Dis = GLOBAL] AND Ev, Dt, Status[i] = 1 for i=1 to n AND Event, Dt, Res[i] = 1 for i=1 to n All events (local or global) are attempted for detection but some event detection results are false i.e these events are not occurred</td>
</tr>
<tr>
<td>P3 EVENT DETECTION FALSE OR INDEFINITE</td>
<td>T10</td>
<td>Some event detections are indefinite, so go for alternative action.</td>
<td>Ev, Type, Dis [string] Ev, Type, Dis [string] Ev, Dt, Status[i] (integer array)</td>
<td>[Ev, Type, Dis = LOCAL OR Ev, Type, Dis = GLOBAL] AND Ev, Dt, Status[i] = 1 for i=1 to n Some events are unnoticed for some reason so Ev, Dt, Status[i] = 1 for i=1 to n</td>
</tr>
<tr>
<td>P7 ALTERNATIVE ACTION</td>
<td>T11</td>
<td>Some condition evaluations are indefinite, so go for alternative action.</td>
<td>[Con, Type, Dis [string] Con, Ev, Status[i] (integer array)</td>
<td>[Con, Type, Dis = LOCAL OR Con, Type, Dis = GLOBAL] AND Con, Dt, Status[i] = 1 for i=1 to n Some condition evaluations are unnoticed</td>
</tr>
</tbody>
</table>
7 Benefit of the work
By this model we clearly can understand the overall scenario of the distributed active database. There are several concurrent situations/operations involved in this work which are very significantly modeled by the Color Petri Net. Since this model is done by Color Petri Net, the modeling language is already attached herewith. The timing constraints modeled by Color Time Petri Net has also significant role to cope up the situation of indefiniteness due to inaccessibility of a remote sub-system.

8 Conclusion
Applicability of CTPN in ‘Distributed Active Database’ has been shown in this paper. Both the primitive and composite events have been considered while we dealt with local and global issues. Concepts of alternative actions in case of ‘event detection being indefinite’ has also been included in the design to address such situation.

References:

[17] Technical Report, Distributed Complex Event Detection, for Pervasive Computing, Dan O’Keeffe, July 2010,Computer Laboratory, University of Cambridge, United Kingdom