Modeling of Updating Moving Object Database Using Timed Petri net Model

F. A.Torky,
Prof. of Computer Science & Engineering and President of Kafer El-Sheekh University, Egypt
Fatorkey@yahoo.com

H. M. Abdulkader,
Lecturer of Information Systems Faculty of Computers and Information Menoufya Univ. Egypt.
Hatem6803@Yahoo.Com

N. Abd El-Wahed,
Dean of Faculty of Computers and Information, Menoufya University, Egypt
Nabil_is48@yahoo.com

Warda El-Kholy
Information Systems Faculty of Computers and Information Menoufya University Egypt
Warda_Elkoly@yahoo.com

Abstract

Tracking moving objects is one of the most common requirements for many location-based applications. The location of a moving object changes continuously but the database location of the moving object cannot update continuously. Modeling of such moving object database should be considered to facilitate study of the performance and design parameters. Such study is essential for selecting the optimal solution in order to minimize the implementation of the overhead cost. Location updating strategy for such type of database is the most important criteria. This paper proposed a timed Petri net model based on one of the most common updating strategies, namely the distance updating strategy. In addition, a method for estimating the time needed to update Moving Object Database (MOD) using the concept of the minimum cycle time in timed Petri nets is presented. This time is the main criterion, which can be used to study the overhead communication cost for MOD. A typical numerical example is given to demonstrate the advantages of proposed modeling technique.

Keywords*: Updating moving object database, Deterministic timed Petri net, Deviation update policy and tracking moving object database.*

1. Introduction

Recent advances in wireless communication systems and Global Position system (GPS) are the main issues that make position tracking of moving objects feasible. Tracking is an enabling technology for many location-based services. As a result, a wide interest of many new applications that depend on location management can be shown in the literature [1], [5]. Tourist services, mobile E-commerce and digital battlefield are examples of these applications [2]. Other application classes that will benefit from tracking include transportation, traffic control, mobile resource management, and mobile workforce. This brought to database researchers the new challenge in the area of MOD.

Traditional database management system (DBMS) is not equipped to handle continuously changing data such as the transient position of moving object. This means that traditional DBMS deals with static data attributes at a given time [3], leading to a rather discrete model. Therefore, in many MOD applications a continuous model for these dynamic objects will be essential in order to mange such moving objects [1], [3], [4]. In this case, an updating strategy for a moving object is required. The objective of this strategy is to accurate track the current location of moving object while minimizing the number of updates. It is obvious that the more often data is updated, the more accurate the data will be. However, the cost of updating data increases with the frequency updating the data. That is, there is a trade-off between updating cost and information accuracy in designing MOD systems. The most common approach is distance update policy, which updates the database every x units of distance. So it provides a certain error in response to a query about the location of any object (e.g.: retrieve the current location of an object?) The answer is within a circle of radius x centered at location L (which is provided in the last updating for database). This approach is used in many applications due to its simplicity [6], [9].

Petri nets are graphical and mathematical modeling tools applicable to many systems. They are promising tools for describing and studying information processing systems that are characterized as being concurrent, asynchronous, distributed, parallel, nondeterministic and or stochastic as a graphical tool [8]. In this paper, a timed Petri net is presented

for updating a MOD. The moving object in this model uses a deviation update policy to update its database location. Then, the Petri net method of the minimum cycle time is applied to estimate the time duration required to update the MOD [8]. The number of moving objects, the number of wireless communication agents, and the number of processors of the DBMS affect this time duration.

The rest of this paper is organized as follows. Moving object database architecture is presented in Section 2. Section 3 describes Petri net basics that are used in this paper. The model for updating the MOD using Petri net is presented in Section 4. Section 5 is an illustration example of the model and how the minimum cycle time method can be used to estimate the updating time. Section 6 discusses some applications of the model. Finally, conclusions and a proposal for future work are drawn in Section 7.

2. Moving object database (MOD) architecture and modeling

As shown in Figure (1) the MOD system modeled in this paper consists of: 1) A number of moving objects each of which is equipped with a GPS receiver, a processor for calculating the deviation of moving object based on deviation updating policy and a local database. 2) A database server with a number of processors, which controls a database for all moving objects, and can be centralized or distributed and 3) wireless agents that provide communication services between moving objects and the DBMS. The history of a moving object's location and time is stored in the database at the database server.

Figure 1. Architecture of the MOD updating system

When the number of available communication agents is limited and there is, more number of moving objects needs to update their location in the central database system. Thus, an overhead results from increasing both the number of update message and the number of wireless communication agents. Therefore, this paper proposed Timed Petri net model to decrease both the number of update messages and overhead of communication cost of the moving object database in an efficient manner. However, the following section explains some basic aspects of Petri net before we introduced the model.

3. Petri net Basics

A Petri net is a graphical and mathematical modeling tool. It consists of three types of object. These objects are places, transitions, and arcs that connect them. In graphical representation, places are drawn as circles, transitions as bars or boxes. Arcs are labeled with their weights (positive integers). Where a K-weighted arc can be interpreted as the set K parallel arcs. Labels for unity weight are usually omitted. Input arcs connect places with transitions, while output arcs start at a transition and end at a place. There are other types of arcs, e.g. inhibitor arcs. Places can contain tokens; the current state of the modeled system (the marking) is given by the number (and type if the tokens are distinguishable) of tokens in each place.

Transitions are active components. They model activities, which can occur (the transition fires), thus changing the state of the system (the marking of the Petri net). Transitions are only allowed to fire if they are enabled, which means that all the preconditions for the activity must be fulfilled (there are enough tokens available in the input places). When the transition fires, it removes tokens from its input places and adds some at all of its output places. The number of tokens removed/added depends on the cardinality of each arc [8], [9], [10]. In modeling using the concept of conditions and events, places represent conditions, and transition represent events. For instance, input (output) places may represent preconditions (post-conditions) to an event (transition). Some typical interpretation of transitions and their input places and output places are shown in Table (1). A formal definition of a Petri net is given in table (2) [8].

Table 1Table 1. Some typical interpretations and places.

Output places	Transition	Input places
Post-conditions	Event	Preconditions
Conclusions	Clause in logic	Conditions
Input signals	Signal processor	Input signals
Buffers	processor	Buffers

Table 2. The definition of a Petri Net.

Formally a Petri net (PN) can be defined as follows
PN = (P ,T, I , O, M0) Where *P = { p1, p2, p3, ... , pm}* is a finite set of places *T = {t1, t2, t3,.....,tn}* is a finite set of transitions where p υ *T≠ Φ, and P∩T = Φ* *I : (P x T) → N* is an input function that defines the directed arcs from places to transitions, and N is a set of nonnegative integer *O : (P x T) → N* is an output function that defines the directed arcs from transitions to places , and *M0: P→ N* is the initial marking.

The classical Petri nets do not include any notion of time; in order to use the Petri net formalism for the quantitative analysis of the performance and reliability of system versus time, a class of Timed Petri net has been introduced. The time delay variables associated with the Petri net can be either deterministic variables (leading to the class of models called deterministic Petri net), or random variables (leading to the class of models called Stochastic Petri net) [8]. When time delay is associated with transitions, this type of net is called timed transition, Petri net [10]. Suppose there is a time delay associated with transition this means that when this transition is enabled tokens remain on the input places of a transition for a time at least equal to the time delay associated with enabled transition before their removal by firing this transition.

4. Model of MOD Updating System Using Timed Petri net

This section presents an application of timed Petri net model. The model of MOD system is shown in Figure (2) and this model consists of three phases:

Phase (1): Moving objects and GPS receivers. Each moving object is equipped with one GPS receiver (for collecting the current real location of the object), one processor (for calculation), and local database (to store the previous location of moving object and a threshold which are used to calculate the deviation of the moving object). The functionality of this part is that moving objects get the information on location and time, and applies distance update policy to generate an updating

message, if the deviation exceeds a specific threshold or if the moving object stops moving, which will be sent to the database server.

Phase (2): Communication Services. This part includes several wireless agents, which provide communication services. The functionality of this part is to provide the communication between moving objects and the database server.

Phase (3): Database server. The information of all moving objects is stored in a database and handled by the DBMS, which is equipped with a number of processors. The main functionality of this part is to update the database with the received messages and generate return messages to update the pervious location of the moving object. The operation of each transition, the tokens in each place and the meaning of each arc inscription in this model shown in Figure (2) are given in Tables 3, 4 and 5 respectively.

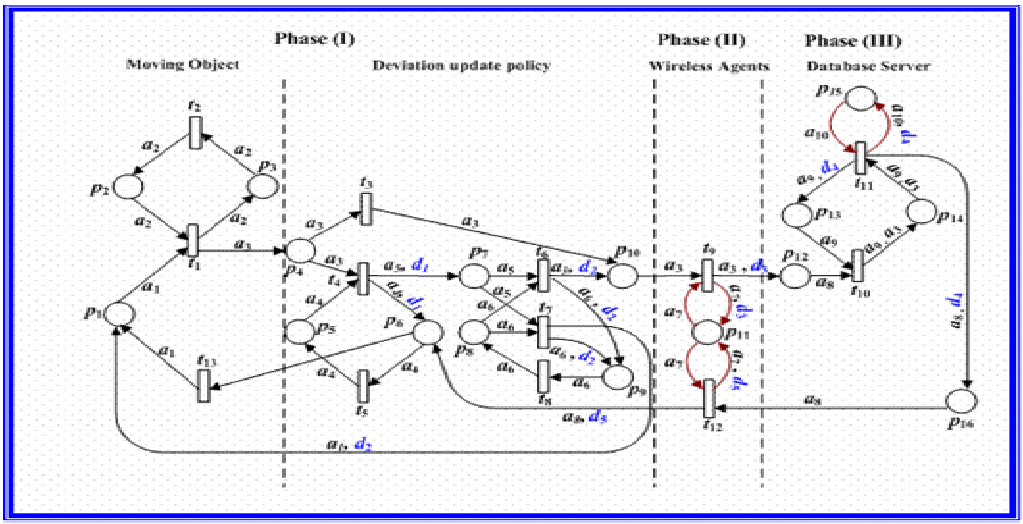

Figure 2. Petri net model for updating moving object database system.

Table 3. The operation of each transition.

Transition	Operation
t_1	A moving object gets its current location and time. Signal from satellites.
t_2	All GPS's receiving the signal from satellites.
t_3	The moving object (mid) stops moving, so sending < mid, cl, ct> to P10.
t_4	Calculating the deviation (did)for each moving object mid.
t_5	Passing all < mid, pl>'s from P6 to P5.
t_6	Comparing did with thid and the result is did > thid.
t_7	Comparing did with thid and the result is did ≤thid.
t_8	Passing all < mid, thid>'s from P9 to P8.
t_9	A wireless agent sends the message <mid, mcl, mct,> from the moving object mid to the database.
t_{10}	A moving object generates a transaction for updating the database.
t_{11}	A database processor executes the transactions for appending the received message <mid, mcl, mct> for the moving object mid.
t_{12}	A wireless agent Updating the location of the previous update for each moving object mid
t_{15}	Moving object gets its previous location and time after updating database

Table 4. The tokens of each place.

Place	Token
P1	<mid> All moving objects, waiting for current location and time. Initially all n moving objects are here.
P2	<gid, gcl, gct>, GPS receivers after collecting the information from satellites.
P3	All GPS's, n , waiting for receiving the signal from satellites
P4	<mid, mcl, mct, >, moving objects with the current location and time waiting for calculation or directly sending to the database server.
P5	<mid, plid>, waiting for sending pl to the moving objects. Initially all locations of the previous update for all n moving object are here.
P6	<mid, plid>, waiting for passing to P5.
P7	The set <mid, did, cl, ct> moving objects with the current location ,current time and deviation
P8	The thresholds for all moving objects, i.e. the set <mid, thid>, waiting for comparison. Initially all threshold values for all n moving object are here.
P9	The thresholds for all moving objects, i.e. the set <mid, thid>, waiting for passing < mid, thid > to P8.
P10	The set <mid , cl, ct>, waiting for sending to the database server.
P11	<wid> wireless agents. Initially there is r = the set of all wireless agents, waiting for receiving the information on moving objects.
P12	<mid, mcl, mct>, the messages of moving objects that are waiting for updating the database.
P13	<db, mid>, waiting for receiving the information on moving objects. Initially all n moving objects are here.
P14	<db, mid, mcl, mct>, transactions waiting for appending the information on the moving objects to the database db.
P15	<pid>, processors. Initially there is K = the set of all k processors managing the database of n moving objects
P16	<mid, cl>, waiting for updating the locations of the previous updates.

5. Illustration Example with Calculation of the Minimum Cycle Time

The minimum cycle time defined as the minimum time required to complete a firing sequence returning to the initial marking after firing each transition at least once [8]. This measure is used only for the timed net. The net shown in Figure (2) can be converted into a timed Petri net shown in Figure (3). A Petri net MATLAB toolbox, available at [12] is used to build the proposed model using MATLAB version 6.5. Since, we can move the delays d_1, d_2, d_3, d_4, d_5 of all the outgoing arcs of t_4, t_6, t_7, t_9, t_{11}, t_{12} to their corresponding transitions. We consider these delays are deterministic and the proposed model is deterministic timed Petri net model. Firings of transitions t_3 and t_4 represent the two cases: one (t_3) for which the moving object stop moving and the other (t_4) for which the moving object continue its motion and need to calculate the deviation. In addition, firings of transitions t_6 and t_7 represent the two cases: one (t_6) for which the deviation exceeds the threshold and the other (t_7) for which the deviation dose not exceed the threshold to simplify the analysis, it is assumed that these two cases occur with equal probabilities. In addition, the Self-loops (t_9-P_{11}, t_{12}-P_{11} *and* t_{11}-P_{15}) in Figure (2) are transformed into the loops as shown in Figure (3).

From studying the structural properties of the net in Figure (3) we can say that this net is bounded, conservative, repetitive and consistent. Now it is easy to find the following:

Incidence Matrix (A)

$$A = A_o - A_i \quad \textbf{(1)}$$

Where A_o = output matrix and A_i = input matrix. The entries of the incidence matrix are defined as follows: $aij = aij + - a_{ij}-$ where $a_{ij}+ = w\ (i, j)$ is the weight of the arc from transition i to its output place j and $a_{ij}- = w\ (i, j)$ is the weight of the arc to transition i from its input place j. When transition t_i fires, a_{ij} + represents the number of tokens deposited on it is output place p_j , a_{ij}- represents the of tokens removed from is input place p_j , a_{ij} represents the change in the number of tokens in place p_j. The following figures show the Petri net MATLAB toolbox to find the incidence matrix of the net mentioned above.

Table 5. The explanation of each arc inscription.

Arcs	Explanation
a1	<mid>, where mid is the identification number of a moving object.
a2	<gid, gcl, gct>, where gid is the identification number of a GPS receiver, gcl is current location of the moving object which has the same identification number as gid, and gct is the time for gcl.
a3	<mid, mcl, mct, >. A moving object mid has the current location and time (mcl, mct).
a4	<mid, pl> A set of locations of the previous update for all moving objects where mid = a moving object, *pl* = the location of the previous update for mid, *id = 1, 2, …, n*}
a5	<mid, did, cl, ct> (where mid = a moving object, did = the Euclidean distance between pl and cl, cl = the current location of mid, and ct = the time point when midis at cl *i = 1, 2, …, n*)
a5, d1	d1 is the time delay for calculation of the deviation of each moving object.
a4, d1	d1 is the time delay for calculation of the deviation of each moving object.
a6	<mid, thid>A set of thresholds for all moving objects, thid=the threshold for mid, *id = 1, 2, …, n}*
a3 ,d2	d2 is the time delay for Comparing did with thid for each moving object.
a7	<wid> The id of wireless communication agent wid
a6 , d2	d2 is the time delay for Comparing did with thid for each moving object.
a3 ,d3	d3 is the time delay associated with wireless communication for sending an update message
a7,d5	d5 is the time delay for wireless communication for sending Updating of the location of the previous update for each moving object mid
a8	<mid, cl> the moving objects current locations send to update the previous location where mid = a moving object ,cl = the current location of mid, *id = 1, 2, …, n*
a9	<db, mid>, where db is the name of the database handling the information of the moving object mid.
a9,a3	<db, mid, mcl, mct, >, a transaction of the database db for updating the current location and time (mcl, mct)
a8,d4	d4 is the time delay for the service provided by the processor pid in the database server that's the time to execute a transaction of the database db.
a10	<pid> The id of processor pid in the database server.
a9,d4	d4 is the time delay for the service provided by the processor pid in the database server, that's the time to execute a transaction of the database db.

Arcs	Explanation
a8,d5	d5 is the time delay for wireless communication for sending Updating of the location of the previous update for each moving object mid

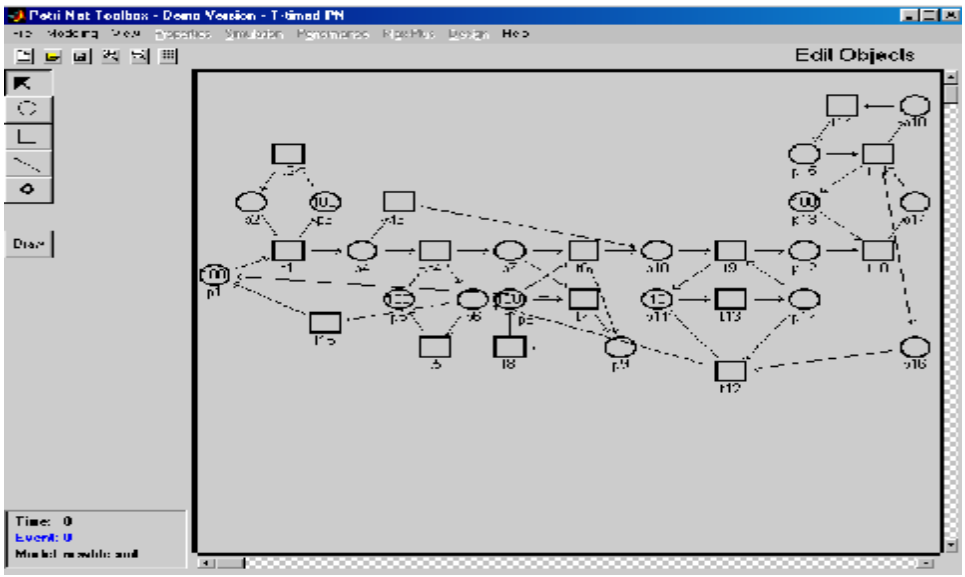

Figure 3. Deterministic timed Petri net model obtained from Figure 2.

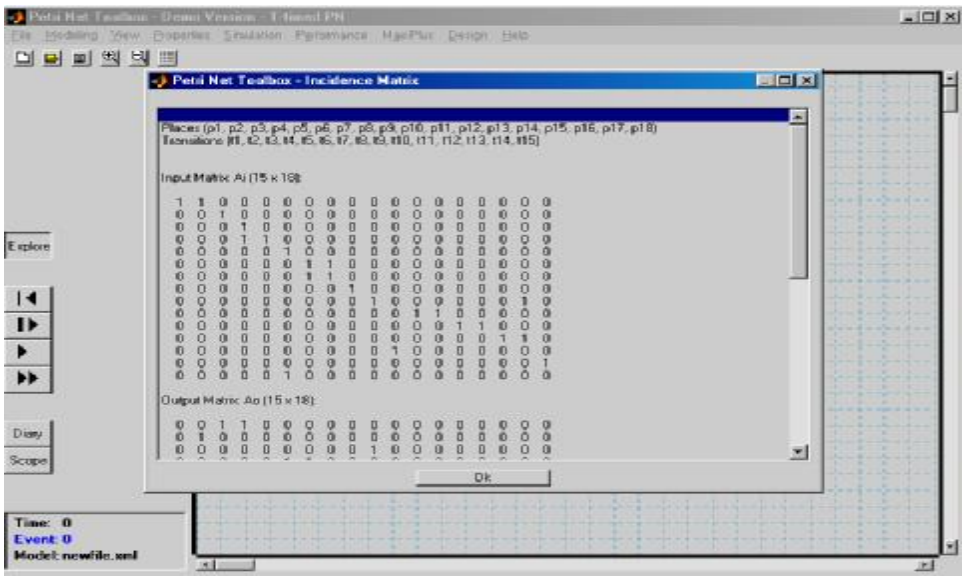

Figure 4-a. Shows the input Matrix of the net shown in Figure 3.

Two concepts are related to the incidence matrix. They are T-invariant and P-invariant, which are useful to find the minimum cycle time [8], [9],[10].

An integer solution x of A^T x = 0 is called T-invariant. The nonzero the corresponding transitions, which belong to a firing sequence transforming a marking M_0 back entries in a T-invariant represent the firing counts of to M_0. The T-invariant indicates the firing sequence transforming M_0 back to M_0 and the number of times these transitions appear in this sequence but does not specify the order of transition firing. In the net of Figure (3) there is a firing sequence from a marking

M back to the same marking M after firing each transition at least once. Such as $<<<t_1, t_1, t_4, t_5, t_7, t_8> t_2, t_1, t_3, t_{13}, t_9, t_{10}, t_{11}, t_{14}, t_{13}, t_{12}, t_{15}> t_2, t_1, t_4, t_5, t_6, t_8, t_{13}, t_9, t_{10}, t_{11}, t_{14}, t_{13}, t_{12}, t_{15}>$. The firing count vector x of this firing sequence is given by:

$$X = (3\ 3\ 1\ 2\ 2\ 1\ 1\ 2\ 2\ 2\ 2\ 2\ 4\ 2\ 2)^T \quad \textbf{(2)}$$

An integer solution y of Ay= 0 is called a P-invariant. This satisfies the following invariant property:

$$y^T M_i = y^T M_0 \quad \textbf{(3)}$$

Where M_0 is the initial marking and Mi is any marking reachable from M_0. The none zero entries in a p-invariant represent weights associated with the corresponding places so that the weight sum of tokens on these places is constant for all markings reachable from an initial marking. There are six (minimum, independent) P- invariants and they are given by:

$$\begin{aligned}
&P_1\ P_2\ P_3\ P_4\ P_5\ P_6\ P_7 P_8\ P_9\ P_{10} P_{11}\ P_{12}\ P_{13}\ P_{14} P_{15}\ P_{16}\ P_{17} P_{18}\\
y_1^T &= (0\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0),\\
y_2^T &= (0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0),\\
y_3^T &= (0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 0),\\
y_4^T &= (0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ 0),\\
y_5^T &= (0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 1) \text{ and}\\
y_6^T &= (1\ 0\ 0\ 1\ 1\ 1\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 0).
\end{aligned} \quad \textbf{(4)}$$

The minimum cycle time can be found by the following equation given in [8]:

$$Minimum\ Cycle\ Time = Max\{(y_k^T (A_i)^T DX / y_k^T M_0\} \quad \textbf{(5)}$$

where, for the net shown in Figure (3), x is the T-invariant given by Equation (2), yk's are six P-invariants given by Equation (4), and Ai = $[aij-]_{n \times m}$, D = the diagonal matrix of di (di is the time delay associated with transition ti , i = 1, 2,…, n) and M_0 = the initial marking are shown below:

$$\begin{aligned}
&P_1\ P_2 P_3 P_4 P_5 P_6 P_7 P_8 P_9 P_{10} P_{11} P_{12} P_{13} P_{14} P_{15} P_{16} P_{17} P_{18}\\
M_0 &= (n\ 0\ n\ 0\ n\ 0\ 0\ n\ 0\ 0\ r\ 0\ n\ 0\ k\ 0\ 0\ 0)^T
\end{aligned}$$

Where n is the number of moving objects, GPS receivers, and tables in the local and server database, r is the number of wireless communication agents, and k is the number of database server processors. $yk^T M_0$ are found as follows:

$$\begin{aligned}
y_1^T M_0 &= n\\
y_2^T M_0 &= n\\
y_3^T M_0 &= r\\
y_4^T M_0 &= n\\
y_5^T M_0 &= k\\
y_6^T M_0 &= 2n
\end{aligned}$$

The input matrix A_i is given by the following matrix:

A_i =

	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}	P_{12}	P_{13}	P_{14}	P_{15}	P_{16}	P_{17}	P_{18}
t_1	+1	+1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
t_2	0	0	+1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
t_3	0	0	0	+1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
t_4	0	0	0	+1	+1	0	0	0	0	0	0	0	0	0	0	0	0	0
t_5	0	0	0	0	0	+1	0	0	0	0	0	0	0	0	0	0	0	0
t_6	0	0	0	0	0	0	+1	+1	0	0	0	0	0	0	0	0	0	0
t_7	+1	0	0	0	0	0	+1	+1	0	0	0	0	0	0	0	0	0	0
t_8	0	0	0	0	0	0	0	0	+1	0	0	0	0	0	0	0	0	0
t_9	0	0	0	0	0	0	0	0	0	+1	0	0	0	0	0	0	+1	0
t_{10}	0	0	0	0	0	0	0	0	0	0	0	+1	+1	0	0	0	0	0
t_{11}	0	0	0	0	0	0	0	0	0	0	0	0	0	+1	+1	0	0	0
t_{12}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	+1	+1	0
t_{13}	0	0	0	0	0	0	0	0	0	0	+1	0	0	0	0	0	0	0
t_{14}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	+1
t_{15}	0	0	0	0	0	+1	0	0	0	0	0	0	0	0	0	0	0	0

The delay matrix D for the net in Figure (3) is the Diagonal matrix given by:

D =

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}	t_{14}	t_{15}
t_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
t_2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
t_3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
t_4	0	0	0	d_1	0	0	0	0	0	0	0	0	0	0	0
t_5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
t_6	0	0	0	0	0	d_2	0	0	0	0	0	0	0	0	0
t_7	0	0	0	0	0	0	d_2	0	0	0	0	0	0	0	0
t_8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
t_9	0	0	0	0	0	0	0	0	d_3	0	0	0	0	0	0
t_{10}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
t_{11}	0	0	0	0	0	0	0	0	0	0	d_4	0	0	0	0
t_{12}	0	0	0	0	0	0	0	0	0	0	0	d_5	0	0	0
t_{13}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
t_{14}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
t_{15}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Thus

$$(A_i)^T\ DX\ =\ (\ 0\ \ 0\ \ 0\ \ 2d_1\ \ 2d_1\ 0\ \ 2d_2\ \ 2d_2\ \ 0\ \ 2d_3\ \ 0\ \ 0\ \ 0\ \ 2d_4\ \ 2d_4\ \ 2d_5\ \ 2d_3+2d_5\ \ 0\ )^T$$

and we find the following for Equation (5):

$$y_1^T(A_i)^T DX / y_1^T\ M_0 = 0$$
$$y_2^T(A_i)^T DX / y_2^T\ M_0 = 4d_2\ /n$$
$$y_3^T(A_i)^T DX / y_3^T\ M_0 = (2d_3+2d_5)\ /r$$
$$y_4^T(A_i)^T DX / y_4^T\ M_0 = 2d_4\ /n$$
$$y_5^T(A_i)^T DX / y_5^T\ M_0 = 2d_4/\ k$$
$$y_6^T(A_i)^T DX / y_6^T\ M_0 = (4d_1\ +2d_2+2(d_3+d_4+d_5))\ /2n$$

From Equation (5) the minimum cycle time of the net in Figure (3) can be given by:

$$The\ \min imum\ cycle\ time\ = Max\ \{0,\ 4d_2\ /n,\ 2(d_3+d_5)\ /r,\ 2d_4\ /n,\ 2d_4/\ k,\ (4d_1\ 2d_2+\ 2(d_3+d_4+d_5)\ /2n\} \qquad \textbf{(6)}$$

6. Application of the proposed Model

The minimum cycle time for the timed net in Figure (3) which is given by Equation (6) corresponds to the minimum time needed to check if the update is necessary for both stopped moving object or the moving object with deviation greater than a specific threshold and, if so, update once for each of n moving objects [11].

For example: consider the net in Figure (3) where we assume that the time delays as follows: d_1= 0.0002 time unit, d_2 = 0.0002 time unit, d_3= 0.01 time unit, d_4 = 1 time unit and d_5 = 0.01 time unit. Also assume that n= 100, r=10 and k=1. From equation (6) the minimum cycle time = $2d_4/k$ = 2 time unit.

If the GPS receiver collects the location information (current location of the moving object and current time) every three-time units, from Equation (6) each object can complete one update through 2 time units, so the system is safe. In other word the system is safe if the GPS receiver collects the location information every t time unit since t > minimum cycle time.

Also according to Equation (6), we can increase $4d_2/n$, $(2d_3+2d_5)/r$, $2d_4/n$ and $(4d_1 + 2d_2 + 2(d_3 + d_4 + d_5))/2n$ up to $2d_4/k$ without affecting the minimum cycle time. For example we can increase $(2d_3 + 2d_5)/r$ to $2d_4/k$ by decreasing the number of wireless agents' r =10 to 1 without affecting the performance of he system. The system with one agent allows each moving object to make its update to the database. This can reduce the cost paid for wireless communication services. Assuming that the wireless communication cost depends only on the time of communication, from Equation (6), we can also increase the number of moving objects without affecting the minimum cycle time. From this, we can deduce that a large number of moving objects can maintain their current locations in the database without needing to increase the number of wireless agents.

Suppose that the GPS receiver collects the location information more frequently, e.g., every 1 time unit, then the above minimum cycle time (2 time units) for each object may be too slow. That is, not all GPS signals can be recorded and the location information of some moving objects may be lost. In this case, it is necessary to decrease $2d_4/k$; by either decreasing d_4 or increasing k. Upgrading or improving the DBMS software so as to speed up transactions in the DBMS can reduce the delay d_4. Adding more DBMS processors can increase the value of k and reduce the minimum cycle time. For the DBMS with more than one processor, the database can be processed in parallel, reducing the time needed for update.

7. Conclusions and Future Work

This paper presents a timed Petri net model for updating moving object database system using distance-updating strategy. In addition, a method for calculation of the minimum cycle time for updating the database is proposed. A Petri net MATLAB toolbox used to study the structural properties of the proposed model. The presented model can be more complex by refining the presented model. For example, transitions t_9 and t_{12} can be refined in order to model wireless communication protocols. Transitions t_{10} and t_{11} also can be expanded to simulate a specific DBMS architecture. Another updating strategy such as a deviation updating strategy can be used instead of distance updating policy.

8. References

[1] O. Wolfson, L. Jiang, P. Sistla, S. Chamberlain, N. Rishe, and M. Deng, "Databases for Tracking Mobile Units in Real Time", Springer-Verlag Lecture Notes in Computer Science 1540, Proceedings of the Seventh International Conference on Database Theory (ICDT), Jerusalem, Israel, , Jan 1999, pp. 169-186.

[2] O. Wolfson, L. Jiang, S. Chamberlain, and S. Dao, "Location Management in Moving Object Databases.", Proceedings of The Second International Workshop on Satellite-Based Information Services (WOSBIS'97), Budapest, Hungary, October 1997.

[3] O. Wolfson , L. .Jiang, S. Chamberlain, and B. Xu, " Moving Object Databases: Issues and Solutions", Proceedings of the 10th International Conference on Scientific and Statistical Database Management (SSDBM98), Capri, Italy, July 1998, pp. 111-122.

[4] O. Wolfson, P. Sistla,, B. Xu, J. Zhou, S. Chamberlain, Y. Yesha, and N. Rishe, " Tracking Moving objects using database technology in DOMINO", proceedings of NGITS 99,the fourth workshop on next generation information technologies and systems, , July 1999, pp. 112-199.

[5] O. Wolfson , S. Chamberlain, , P. Sistla, B. Xu, and J. Zhou, "DOMINO: Databases fOr MovINg Objects tracking,", the Proceedings of the ACM-SIGMOD 1999, International Conference on Management of Data, Philadelphia, PA,

June 1999.
[6] QUALCOMMInc. Retrieved September 4, 2005, from: // http://www.qualcomm.com
[7] At Road Inc. Retrieved August 20, 2005, From: // http://www.atroad.com/
[8] T. Murata, "Petri Nets: Properties, Analysis and Application ", Proceedings of the IEEE, Vol. 77, No. 4, April 1989, pp. 541-580.
[9] O. Wolfson, and H. Yin, "Accuracy and resource consumption in tracking and location prediction", proceedings of 8Th international symposiums on spatial and temporal databases, July 2003. pp. 325-343.
[10] R. Zurawski, and M. Zhou, " Petri Nets and Industrial Application: A Tutorial", IEEE transaction on industrial electronics, VOL.41, NO.6, December 1994.
[11] T. Murata, J. Yim, H. Yin, and O. Wolfson " Fuzzy-Timing Petri-Net Model for Updating Moving Objects Database" Proceedings of the 2003 VIP Scientific Forum of International Conference on IPSI (Internet, Processing, Systems, and Interdisciplinaries), Sveti Stefan, Montenegro, Yugoslavia, October 2003, pp.1-7.
[12] C. Mahullea, M. Hanako, and O. Pasttravanu, Petri Net Toolbox for MATLAB. Retrieved August 20, 2005,
[13] From: // http://www.ac.tuiasi.ro/pntool .