Modelling of Updating Moving Object Database Using Timed Petri Net Model

Hatem Abdul-Kader and Warda El-Kholy
Faculty of Computers and Information, Menoufya University, Egypt

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. Modelling of such moving object database should be considered to facilitate study of the performance and design parameters for this database feature. 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 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 moving object database. A typical numerical example is given to demonstrate the advantages of proposed modelling technique.*

Keywords: *Updating moving object database, deterministic timed Petri net, deviation update policy, tracking moving object database*.

Received June 15, 2008; accepted September 3, 2009

1. Introduction

Recent advances in wireless communication systems and Global Position System (GPS) are the main issues that make position tracking of moving objects is 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], and [6]. Tourist services, mobile E-commerce and digital battlefield are examples of these applications [4]. Other application classes that will benefit from tracking include transportation, traffic control, mobile resource management, and mobile workforce. This brought to database researchers a new challenge in the area of Moving Object Database (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 [6], leading to a rather discrete database model. Therefore, in many MOD applications a continuous model for these dynamic objects will be essential in order to mange such moving objects [5, 6, 3]. 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 centred at location L (which is provided in the last updating for database). This approach is used in many applications due to its simplicity [8, 5].

Petri nets are graphical and mathematical modelling 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 [3]. 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 minimum cycle time method of Petri net is used to estimate the required time to update the MOD [3, 4]. 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 section 2. Section 3 describes the Petri net basics. The model for updating the MOD using Petri net is presented in section 4. Section 5 gives an illustration

example of the proposed model and how the minimum cycle time technique can be used to estimate the updating time. Section 6 discusses some applications of the proposed model. Finally, conclusions and a proposal for future work are drawn in section 7.

2. Moving Object Database Architecture and Modelling

The architecture of MOD system which will be modelled in this paper consists of:

- 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.
- A database server with a number of processors, which controls a database for all moving objects, and can be centralized or distributed.
- 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. The component of this architecture is shown in Figure 1.

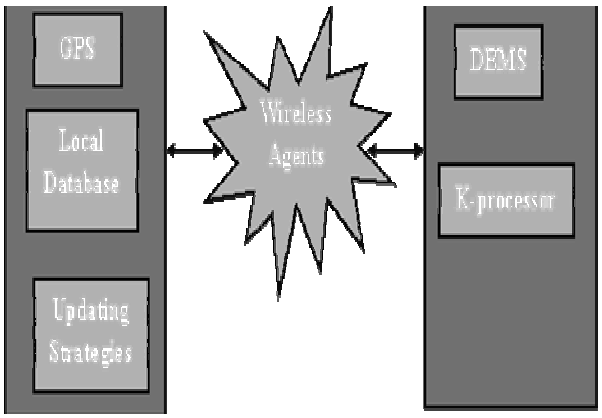

Figure 1. Architecture of the MOD updating system.

In the MOD model, 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.

3. Petri Net Basics

A Petri net is a graphical and mathematical modelling 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 labelled with their weights (positive integers). Where a *K*-weighted arc can be interpreted as the set of *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. Places can contain tokens; the current state of the modelled 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 [4, 3]. In modelling using the concept of conditions and events, places represent conditions, and transitions 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 [3].

Table 1. Some typical interpretations and places.

Input places	Transition	Output places
Preconditions	Event	Post-conditions
Conditions	Clause in logic	Conclusions
Input signals	Signal processor	Output 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 U *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) [3]. When time delay is associated with transitions, this type of net is called timed transition, Petri net [4]. 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 for MOD. This model system is shown in Figure 2, which consists of three items:

- 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.
- 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.
- 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 of Figure 2 model are given in Tables 3, 4 and 5, respectively.

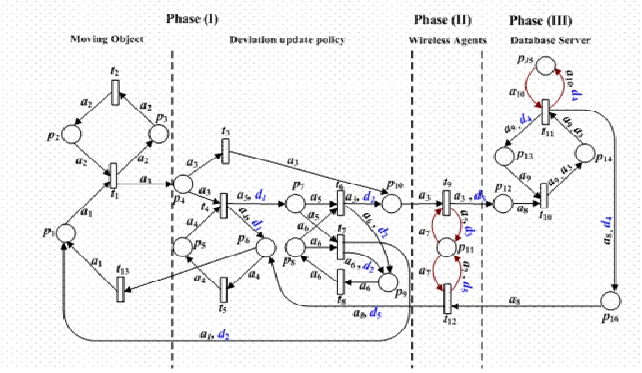

Figure 2. Petri net model for updating MOD system.

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 is defined as the minimum time required completing a firing sequence returning to the initial marking after firing each transition at least

once [3]. This measure is used only for the timed net. Figure 2 net can be converted into a timed Petri net as shown in Figure 3. A Petri net MATLAB toolbox which was available at [2] is used to build the proposed model. Since, we can move the delays *d1*, *d2, d3, d4, d5* 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. Firings of transitions t_3 and t_4 give two different cases. The first one uses t_3 for the moving objects which stop their moving, while in the second case uses t_4 for the moving objects which continue their motion and need to calculate the deviation. In addition, firings of transitions t_6 and t_7 give two different cases. The first one uses t_6 for the moving objects which their deviation exceeds the threshold while the second case uses t_7 for the moving objects which their deviation does 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-P11, t_{12}-P11 and t_{11}-P15)* as shown in Figure 2 are transformed into the loops as shown in Figure 3. From studying the structural properties of Figure 3 net, we can say that this net is bounded, conservative, repetitive and consistent. Also, the incidence matrix A of this net can be written as follows [3]:

$$A = A_o - A_i \quad (1)$$

where A_0 output matrix and Ai is is the input matrix. The entries of the incidence matrix are defined as follows: $a_{ij} = a_{ij}^{+} - a_{ij}^{-}$ where $a_{ij}^{+} = w\ (i, j)$ is the weight of the arc from transitions *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. Figure 4 shows the resulted incidence matrix from the Petri net MATLAB toolbox of Figure 3 net.

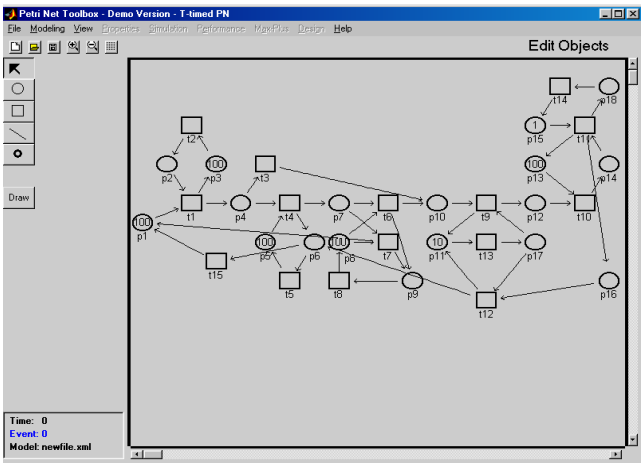

Figure 3. Deterministic timed Petri net model obtained from Figure 2.

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.

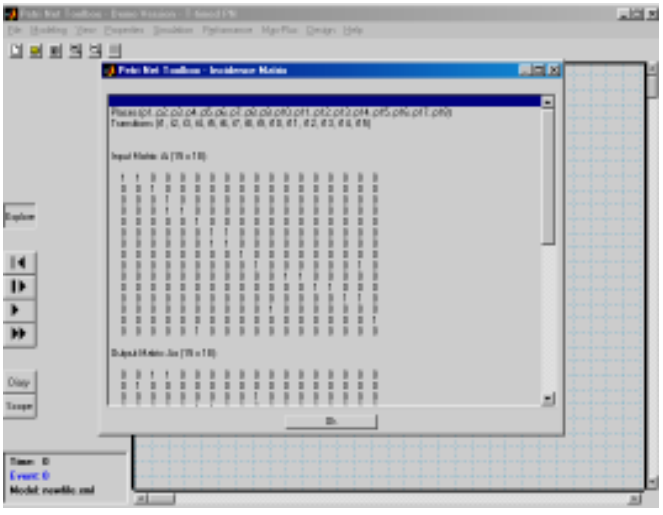

Figure 4. The input matrix of Figure 3 net.

Two concepts related to the incidence matrix are especially useful in studying properties of Petri net models and finding the minimum cycle time. They are

T-invariant, and *P-invariant* [4, 3]. An integer solution X of $A^TX=0$ is called a *T-invariant*. The nonzero entries in a *T-invariant* represent the firing counts of the corresponding transition that belong to a firing sequence transforming a marking M_0 back to M_0. Although a *T-invariant* states the transitions comprising the firing sequence transforming a marking M_0 back to M_0, and the number of times these transitions appear in this sequence, it does not specify the order of transitions firings. An integer solution Y of $AY=0$ is called a *P-invariant*. The *P-invariant* can be explained intuitively in the following way. The nonzero entries in a *P-invariant* represent the weights associated with the corresponding places so that the weighted sum of tokens on theses places is constant for all markings reachable from an initial marking. In 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 $<<<t_2, t_1, t_4, t_5, t_7, t_8>t_2, t_1, t_3, t_{13}, t_9, t_{10}, t_{11}, t_{14}, t_{13}, t_{12}, t_{15}>t_2, t_1, t_4, t_5, t_6, t_8, t_{13}, t_9, t_{10}, t_{11}, t_{14}, t_{13}, t_{12}, t_{15}>$. The firing count vector x of this firing sequence is given by:

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

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

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

where M_0 is the initial marking and M_i 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 which are given by:

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

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

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

where

$(X)_{n\,x1}$ is the T-invariant.

$(Y_k)_{m\,x1}$ is the P-invariants.

$(A_i)_{n\,x\,m}$ is the input matrix.

(d_i) is the time delay associated with transition t_i, $i = 1, 2, \ldots, n$.

$(D)_{n\,x\,n}$ is the diagonal matrix of d_i.

$(M_0)_{m\,x\,1}$ is the initial marking.

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

where n is the number of moving objects, r is the number of wireless communication agents, and k is the number of database server processors. Therefore, $Y_k^T M_0$ 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	+1	+1	0	0	0	0	0	0
t_{11}	0	0	0	0	0	0	0	0	0	0	0	0	+1	+1	0	0	0	0
t_{12}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	+1	+1	0	0
t_{13}	0	0	0	0	0	0	0	0	0	+1	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	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 of Figure 3 net is a diagonal matrix which is 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

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

Finally from equation 5, the minimum cycle time of Figure 3 net can be given by:

$$\text{The minimum cycle time} = \text{Max}\{0, 4d_2/n, 2(d_3+d_5)/r, 2d_4/n, 2d_4/k, (4d_1+2d_2+2(d_3+d_4+d_5))/2n\} \quad (6)$$

6. Application of the Proposed Model

The minimum cycle time for the timed net of Figure 3 is given by equation 6. This is corresponding 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 [4]. For example, in Figure 3 net it is assumed that the time delays in time units is given as follows: *d1= 0.0002* , *d2 = 0.0002* , *d3= 0.01*, *d4 = 1* and *d5 = 0.01* . Also it is assumed that *n= 100*, *r=10* and *k=1*. Finally in equation 6, it is assumed that the minimum cycle time is given *2d4/k = 2* time unit.

If the GPS receiver collects the location information every three-time units then each object can complete one update through *2* time units, so the system is safe. In other words, the system is safe if the GPS receiver collects the location information every *t* time unit since *t* > minimum cycle time.

Moreover, according to equation 6, the following elements can be increased *4d2/n*, *(2d3+2d5)/r, 2d4/n* and *(4d1 + 2d2 + 2(d3 + d4 + d5))/2n* from their values to 2d4/k without affecting the minimum cycle time. For example we can increase *(2d3 + 2d5) /r* to *2d4/k* by decreasing the number of wireless agents from *r =10* to *r=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 paid cost for wireless communication services. 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 *2d4/k;* by either decreasing d4 or increasing *k*. Upgrading the DBMS software so as to speed up transactions in the DBMS which can reduce the delay *d4*. Adding more DBMS processors can increase the value of *k* consequently reducing 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. This model is developed based on 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 is used to study the structural properties of the proposed model. The presented model can be more complex by refining some items in the model structure. For example, transitions t_9 and t_{12} can be refined in order to model other wireless communication protocols. Transitions t_{10} and t_{11} also can be expanded to simulate a specific DBMS architecture. Moreover, other updating strategy such as a deviation updating strategy can be used instead of distance updating policy.

References

[1] Hartmut R. and Schneider M., *Moving Object Databases*, Morgan Kaufmann Publishers, USA 2005.

[2] Mahullea C., Hanako M., and Pasttravanu O., "The Timed Petri-Net Simulator," *Computer Journal of Petri Net Toolbox for MATLAB*, vol. 15, no. 3, pp. 211-228, 2006.

[3] Murata T., "Petri Nets: Properties, Analysis and Application," *in Proceedings of the IEEE*, Japan, pp. 541-580, 1989.

[4] Murata T., Yim J., Yin H., and Wolfson O., "Petri-Net Model and Minimum Cycle Time for Updating a Moving Objects Database," *The International Journal of Computer Systems Science and Engineering*, vol. 21, no. 3, pp. 207-213, 2006.

[5] QUALCOMM Inc, www.qualcomm.com, 2006.

[6] Wolfson O., Chamberlain S., Sistla P., Xu B., and Zhou J., "Domino: Databases for Moving Objects Tracking," *in Proceedings of the ACM-SIGMOD International Conference on Management of Data*, Philadelphia, pp. 547-549, 1999.

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

[8] Wolfson O. and Yin H., "Accuracy and Resource Consumption in Tracking and Location Prediction," *in Proceedings of 8th International Symposiums on Spatial and Temporal Databases*, Greece, pp. 325-343, 2003.

[9] Wolfson O., Jiang L., Sistla P., Chamberlain S., Rishe N., and Deng M., "Databases for Tracking Mobile Units in Real Time," *in Proceedings of*

the 7th International Conference on Database Theory, Jerusalem, pp. 169-186, 1999.

[10] Wolfson O., Jiang L., Chamberlain S., and Dao S., "Location Management in Moving Object Databases," *in Proceedings of The 2ed International Workshop on Satellite-Based Information Services,* Hungary, pp. 422-432, 1997.

[11] Wolfson O., Sistla P., Xu B., Zhou J., Chamberlain S., Yesha Y., and Rishe N., "Tracking Moving Objects Using Database Technology in Domino," *in Proceedings of the 4th Workshop on Next Generation Information Technologies and Systems,* Israel, pp. 112-199, 1999.

[12] Zurawski R. and Zhou M., "Petri Nets and Industrial Application: A Tutorial," *in Proceedings of the IEEE Transactions on Iindustrial Electronics*, New Jersey, pp. 567-583, 1994.

Hatem Abdul-Kader obtained his BS and MSC, both in electrical engineering from the Alexandria University, Faculty of Engineering, Egypt, 1990 and 1995, respectively. Obtained his PhD degree in electrical engineering also from Alexandria University, Faculty of Engineering, Egypt in 2001, specializing in neural networks and its applications. He is currently a lecturer in the Information Systems Department, Faculty of Computers and Information, Menoufya University, Egypt, since 2004.

Warda El-Kholy obtained her BSc in 2002 from the Electronic Engineering Department of Computer Science and Engineering, Faculty of Electronic Engineering, Menoufya University. In 2007, she obtained her MSc of computers and information from Information Systems Department, Faculty of Computers and Information, Menoufya University. Currently, she is pursuing her PhD from the Faculty of Engineering and Computer Science, Concordia University, Canada.