Petri-Net Model and Minimum Cycle Time for Updating Moving Objects Database[1]

Tadao Murata, Jaegeol Yim, Huabei Yin, and Ouri Wolfson

Department of Computer Science (M/C 152), 851 S. Morgan, Chicago, IL 60607, USA
{murata, jyim, wolfson, hyin}@cs.uic.edu

Abstract. Moving objects database (MOD) contains the information (locations and times) of moving objects, e.g., taxicabs, trucks, airplanes, polar bears, etc. This information continuously changes and thus needs to be updated frequently in order to maintain the accuracy and usefulness of the information. This paper presents a fuzzy-timing Petri net model for updating a MOD and a method for estimating the time needed to update its database using the concept of the minimum cycle time of a Petri net. It is expected that the results of this paper are useful to study MOD design issues, such as the trade-off between update cost and information accuracy, and how often the database can be updated, given the number of moving objects, MOD system resources, delays of various MOD operations, etc.

1. Introduction

Recent advances in wireless communication and miniaturised computing and se nsor devices have brought to database researchers the new challenge in the are a of moving objects databases (MOD) [1]. MOD applications include commerci al systems for mobile fleet management such as taxicabs and trucks, as well a s in the military in the context of the digital battlefield [2]. For example, for a MOD representing the locations of taxicabs, a typical query may be: “retriev e the free cabs that are currently within one mile from a customer's location”. For a MOD containing the current positions of moving objects in a battlefield, a typical query may be: “retrieve the friendly helicopters within a given regio n”. Unlike the traditional database where data changes only when updated, the actual data for MOD is changing continuously. It is impossible or impractical to update data in MOD continuously. It is obvious that the more often data is updated, the more accurate the data is. However, the cost of updating data increases as more frequently the data is updated. That is, there is the trade-off between update cost and

[1] This work was supported by NSF Grants, CCR-9988326, ITR-0086144, NSF-0209190, CCR-0070738, and EIA-0000516

information accuracy in designing MOD systems.

To reduce the update cost, there have been proposed many strategies/policies to update less frequently. In the literature of wireless networks, update policies are classified by Bar-Noy, et al [3] into the following three categories: the *time-based policy,* the *movement-based policy*, and the *distance-based policy*. In the *time-based update policy*, the location of each moving object is updated periodically every T units of time. In the *movement-based update policy*, the location is updated after the moving object has moved out of certain boundaries. Finally in the *distance-based update policy*, the location is updated when the distance between the last (stored) location and the current (actual) location exceeds a predefined value (threshold). In the literature of road networks, Wolfson et al [2] proposed a set of *dead-reckoning policies* when the route or the destination of the trip is available, which update the database location whenever the distance between the current real location and the expected location exceeds a given threshold. When the route and destination of the trip are not given, two update policies in [4] can be applied. One is the *deviation policy* that works in a way similar to the dead reckoning policies, and the other is the *distance policy* that works in a way similar to the distance-based update policy in wireless networks.

This paper presents a fuzzy-timing Petri net model for updating a MOD. Our model uses a distance-based update policy. Otherwise, it is independent of the specific update policies mentioned above. The MOD system modeled in this paper consists of 1) a number of moving objects each of which is equipped with a Global Positioning System (GPS) receiver, a processor for calculation and a wireless device for communication; 2) a database management system (DBMS) with a number of processors, which control a database for all moving objects, and can be centralized or distributed; and 3) wireless agents[2] 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 DBMS.

A moving object could be any one of the following objects that move around on the surface of the earth or in the air: a vehicle, a cell phone, an airplane, a ship, etc. The GPS receiver equipped on the moving object communicates with the satellite periodically to obtain specially coded satellite signals and manipulates them into the position and the time of transmission.

The satellite signals are provided by the US Department of Defense (DOD) for free. Since the satellites broadcast their signals, any number of GPS receivers can receive the signals at the same time.

At this moment, only r moving objects can be in one-to-one mapped with r wireless communication agents, where r is the total number of available communication agents. Thus only r (out of the n) moving objects can send their information in parallel. If more moving objects have to communicate with the control center at the same time, we have to increase the number of wireless communication agents.

[2] No particular wireless agents or particular wireless communication services are implied here.

The information (location and time) sent to the DBMS is buffered at its storage device as transactions. The DBMS repetitively picks up one transaction from the buffer and executes it. In order for the DBMS to update the database, it has to read the table from the secondary storage into the main memory. Accessing a secondary storage takes a relatively long time. If updating the database takes too long time, then it may be necessary to improve the DBMS by increasing the number of processors, changing disk systems, or upgrading the hardware system and/or software systems.

The purpose of this paper is to present a fuzzy-timing high-level Petri net (FTHN) model [5] for updating the MOD system described above. Then, the Petri net method of the minimum cycle time [6] is applied to estimate the time duration required to update the MOD. This time duration is affected by the number of moving objects, the number of wireless communication agents, and the number of processors of the DBMS, etc.

The rest of this paper is organized as follows. A fuzzy-timing high-level Petri-net (FTHN) model for updating the MOD is presented in Section II. Section III describes how the minimum cycle time method can be used to estimate updating time. Section IV discusses some applications of the model. Finally, we conclude the paper and propose the future work in Section V.

2. Petri Net Model of a MOD System

The concept and method of the FTHN were first introduced in [5]. Since then, they have been applied to many areas [7, 8]. This paper presents another application of the FTHN, namely to MOD systems. To economy of space, it is assumed that the reader has some knowledge on the method of the FTHN modelling discussed in [5, 8].

The high-level Petri net shown in Fig. 1 is a general FTHN model for updating a MOD. This model consists of the following three parts:

1) *Part I. 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 one wireless communication device (for communication with the DBMS). The functionality of this part is that moving objects get the information on location and time, and applies a certain update policy to generate an updating message, which will be sent to the DBMS.
2) *Part II. Communication Services*. This part includes serval wireless agents which provide communication services. The functionality of this part is to send the messages generated by moving objects to the DBMS.
3) *Part III. Database Management System* (*DBMS*). The information of all moving objects is stored in a database and handled by the DBMS, which is equipped with a number of processors. It can be either centralized in the case of road networks, or distributed in the case of cellular wireless networks. The main functionality of this part is to update the database with the received messages and generate return messages with the information the objects want.

The color set (attributes) of tokens in each place, the operation/role of each transition and the meaning of each arc inscription in the FTHN model shown in Fig. 1 are given in Tables I, II and III, respectively. The initial marking M_0 of the net in Fig. 1 is given as follows: $M_0(P_1) = M_0(P_8) = \sum M$ = the set of n moving objects = $\{mid \mid mid = 1, 2, \ldots, n\}$; $M_0(P_3) = \sum G$ = the set of n GPS receivers = $\{gid \mid gid = 1, 2, \ldots, n\}$; $M_0(P_6) = \sum R$ = the set of r wireless agents = $\{wid \mid wid = 1, 2, \ldots, r\}$; and $M_0(P_{10}) = \sum K$ = the set of k processors in the DBMS = $\{pid \mid pid = 1, 2, \ldots, k\}$; and $M_0(P_i) = \Phi$ for the rest of places P_i.

Table 1. Tokens of each place

Places	Color Set of Token
P_1	<*mid, th*>, moving objects waiting for current location and time. Initially all n moving objects in $\sum M$, are here.
P_2	<*gid, gcl, gct*>, GPS receivers after collecting the information from satellites.
P_3	{<*gid*>}, GPS receivers. Initially there is $\sum G$ = the set of all n GPS receivers waiting for the signals from satellites.
P_4	<*mid, mcl, mct, th*>, moving objects with the current location and time.
P_5	<*mid, mcl, mct, th*>, moving objects that are to send their current locations and times to the database.
P_6	{<*wid*>}, wireless agents. Initially there is $\sum R$ = the set of all r wireless agents, waiting for receiving the information on moving objects.
P_7	<*mid, mcl, mct, th*>, the messages of moving objects that are waiting for updating the database.
P_8	<*db, mid*>, waiting for receiving the information on moving objects. Initially all n moving objects in $\sum M$, are here.
P_9	<*db, mid, mcl, mct, th*>, transactions waiting for appending the information on the moving objects to the database *db*.
P_{10}	{<*pid*>}, processors. Initially there is $\sum K$ = the set of all k processors managing the database of n moving objects
P_{11}	<*mid, mreinfo*>, moving objects with return information[3].

[3] The return information is determined by a particular update policy used.

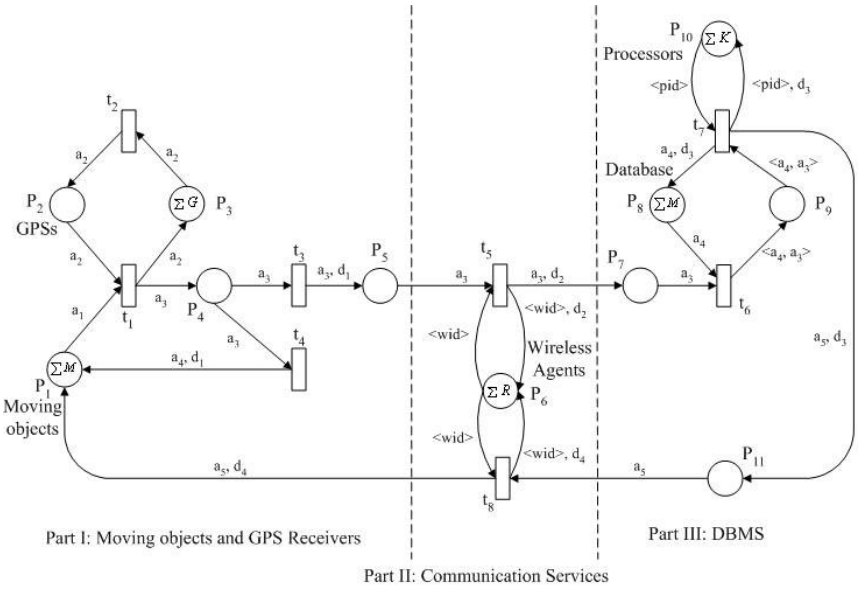

Fig. 1. High-level Petri net model for updating moving objects database

Table 2. Operation of each transition

Transition	Operation
t_1	A moving object gets its current location and time.
t_2	A GPS receiver receives the current location and time signal from satellites.
t_3	A moving object checks if it can generate an update at current time and location by applying a particular update policy[4].
t_4	A moving object checks the opposite case of t_3.
t_5	A wireless agent sends the message <*mid, mcl, mct, th*> from the moving object *mid* to the database.
t_6	A moving object generates a transaction for updating the database.
t_7	A database processor executes the transactions for appending the received message <*mid, mcl, mct*> for the moving object *mid*.
t_8	A wireless agent sends the message <*mid, mreinfo*> from the database to the moving object *mid*.

[4] Transitions t_3 and t_4 may involve operations and information which are not modeled in this paper, and can be refined according to a particular update policy used.

Table 3. Explanation of each arc inscription

Arcs	Explanation
a_1	<*mid, th*>, where *mid* is the identification number of a moving object, and *th* is the threshold.
a_2	<*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*.
a_3	<*mid, mcl, mct, th*>. A moving object *mid* has the current location and time (*mcl, mct*) and threshold *th*.
a_3, d_1	d_1 is the time delay for checking if a moving object *mid* can generate an update message at the current location and time (*mcl, mct*) with the threshold *th*.
a_3, d_2	d_2 is the time delay associated with wireless communication for sending an update message a_3.
a_4	<*db, mid*>, where *db* is the name of the database handling the information of the moving object *mid*.
<a_4, a_3>	<*db, mid, mcl, mct, th*>, a transaction of the database *db* for updating the current location and time (*mcl, mct*) and threshold *th* of the moving object *mid*.
a_4, d_1	d_1 is the time delay for checking if a moving object *mid* cannot generate an update message at the current location and time (*mcl, mct*) with the threshold *th*.
a_4, d_3	d_3 is the time delay for the service provided by the processor *pid* in the DBMS, that's the time to execute a transaction of the database *db*.
a_5	<*mid, mreinfo*>, where *mreinfo* is the return information of the moving object *mid*.
a_5, d_3	d_3 is the time delay for the service provided by the processor *pid* in the DBMS, that's the time to execute a transaction of the database *db*.
<*wid*>	The id of wireless communication agent *wid*
<*wid*>, d_2	d_2 is the time delay associated with wireless communication for sending an update message.
<*wid*>, d_4	d_4 is the time delay for wireless communication for sending an return message.
<*pid*>	The id of processor *pid* in the DBMS.
<*pid*>, d_3	d_3 is the time delay for the service provided by the processor *pid* in the DBMS.

3. Minimum Cycle Time

The minimum cycle time [6] defined as the minimum time required to fire each transition at least once and returned to the initial making in a timed Petri net. A method of finding the minimum cycle time of a timed net is given in [6]. Using this method, we will find the minimum cycle time of the net shown in Fig. 1. Since the method is for a timed net, we unfold the high-level net in Fig. 1 into a timed (colorless) net shown in Fig. 2.

Each of all the outgoing arcs of t_3, t_4, t_5, t_7 and t_8 are labelled with delays, d_1, d_1, d_2, d_3, and d_4, respectively. Thus, we can move these delays from the arcs to their corresponding transitions. Each of the delays d_1, d_2, d_3 and d_4 can be specified either by constants (crisp numbers) or fuzzy numbers. Firings of transitions t_3 and t_4 represent the two cases: one (t_3) for which the difference[5] exceeds the threshold and the other (t_4) for which the difference does not exceed the threshold. The self-loops (t_5-P_6, t_8-P_6 and t_7-P_{10}) in Fig. 1 are transformed into the loops, so that we can use the incidence matrices of Petri nets [5]. Thus unfolding the high-level net of Fig. 1 results in the colorless timed net shown in Fig. 2. It is easy to find the following incidence matrix A [6] of the net in Fig. 2.

$$A = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \\ t_8 \\ t_9 \\ t_{10} \end{array} \begin{array}{c} \begin{array}{cccccccccccc} P_1 & P_2 & P_3 & P_4 & P_5 & P_6 & P_7 & P_8 & P_9 & P_{10} & P_{11} & P_{12} & P_{13} \end{array} \\ \begin{bmatrix} -1 & -1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 \end{bmatrix} \end{array}$$

Let A be the incidence matrix [5] of a Petri net. Vectors (solutions) x and y of $A^T x = 0$ and $A y = 0$ are called the T-invariant and S-invariant of the net, respectively. A T-invariant x of nonnegative integers is a firing count vector, each entry of which represents the number of firings in a firing sequence from a marking M back to the same marking M, if there exists such a sequence. In the net of Fig. 2, there is a firing sequence from a marking M back to the same marking M after firing each transition at least once, such as $<<t_2, t_1, t_4>^m, t_2, t_1, t_3, t_9, t_5, t_6, t_7, t_{10}, t_9, t_8>$, where *m* is the number

[5] The difference is computed by a specific update policy. It can be either the time difference for *time-based* policy, or the number of the cells crossed for *movement-based* policy, or the distance difference for *distance-based* policy and *dead-reckoning* policies.

of times that update was not needed per each update needed, and determined based on a specific update policy and specific MOD. Since transitions t_1 and t_2 fire m+1 times, t_4 fires m times, t_9 fires twice, and each of the rest fires once, the firing count vector x of this firing sequence is given by:

$$x = (m+1\ m+1\ 1\ m\ 1\ 1\ 1\ 1\ 2\ 1)^T \quad (\mathbf{1})$$

On the other hand, S-invariants y (solutions y of A y = 0) satisfies the following invariant property:

$$y^T M_i = y^T M_0 \quad (\mathbf{2})$$

where M_0 is the initial marking, and M_i is any marking reachable from M_0. Equation (2) implies that the (weighted) sum of tokens in places p such that y(p) > 0 is invariant [6]. It is easy to verify that there are five (minimum, independent) S-invariants of the net in Fig. 2, and they are given by:

$$\begin{array}{l} \quad P_1\ P_2\ P_3\ P_4\ P_5\ P_6\ P_7\ P_8\ P_9\ P_{10}\ P_{11}\ P_{12}\ P_{13} \\ y_1^T = (0\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0), \\ y_2^T = (0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 0), \\ y_3^T = (0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ 0), \\ y_4^T = (0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 1), \\ y_5^T = (1\ 0\ 0\ 1\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 0). \end{array} \quad (\mathbf{3})$$

Each of these five S-invariants has the physical meaning described in TABLE IV.

Table 4. Physical meaning of each S-invariant

S-invariant	Physical Meaning
y_1	GPS receivers are either in place P_2 or P_3 and their number n is invariant.
y_2	Wireless communication agents are either in place P_6 or P_{12}.and their number r is invariant.
y_3	Records (tables) of n moving objects in the database are either in place P_8 or P_9 and their number n is invariant.
y_4	DBMS processors are either in place P_{10} or P_{13}.and their number k is invariant.
y_5	Moving objects are in place P_1, P_4, P_5, P_7, P_9, or P_{11}.and their number n is invariant.

The minimum cycle time can be found by the following Equation (4) given in [6]:

$$\text{Minimum Cycle Time} = Max\ \{y_k^T (A^-)^T D x / y_k^T M_0\} \quad (\mathbf{4})$$

where, for the net shown in Fig. 2, x is the T-invariant given by (1), y_k's are the five S-invariants given by (3), and $A^- = [a_{ij}^-]_{n \times m}$, with $a_{ij}^- = w(P_j, t_i)$, D = the diagonal matrix of delays d_i, i = 1, 2, …, n; and M_0 = the initial marking shown below:

$P_1\ P_2\ P_3\ P_4\ P_5\ P_6\ P_7\ P_8\ P_9\ P_{10}\ P_{11}\ P_{12}\ P_{13}$

$M_0 = (n\ 0\ n\ 0\ 0\ r\ 0\ n\ 0\ k\ 0\ 0\ 0)$,

where n = the number of moving objects = the number of GPS receivers = the number of tables in the database, r = the number of wireless communication agents, and k = the number of DBMS processors.

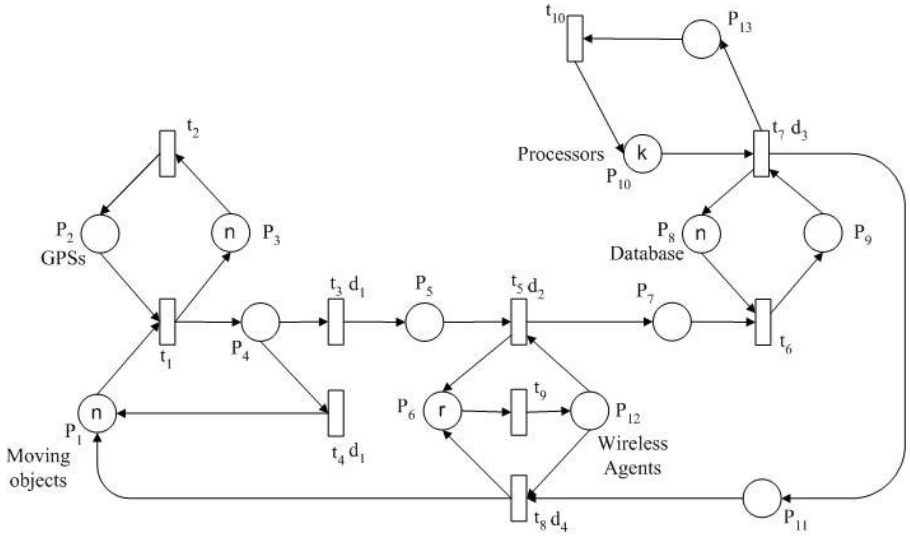

Fig. 2. Timed net model obtained from Fig. 1.

For the net in Fig. 2, the denominators of Equation (4) are found as follows:

$y_1^T M_0 = n$,

$y_2^T M_0 = r$,

$y_3^T M_0 = n$,

$y_4^T M_0 = k$,

$y_5^T M_0 = n$.

The incidence matrix A^- of the net in Fig. 2 is given by the following matrix:

$$A^- = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \\ t_8 \\ t_9 \\ t_{10} \end{array} \begin{array}{c} \begin{array}{cccccccccccc} P_1 & P_2 & P_3 & P_4 & P_5 & P_6 & P_7 & P_8 & P_9 & P_{10} & P_{11} & P_{12} & P_{13} \end{array} \\ \begin{bmatrix} +1 & +1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & +1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & +1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & +1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & +1 & 0 & 0 & 0 & 0 & 0 & +1 & 0 \\ 0 & 0 & 0 & 0 & 0 & +1 & +1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 & +1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 & +1 & 0 \\ 0 & 0 & 0 & 0 & 0 & +1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 \end{bmatrix} \end{array}$$

The delay matrix D for the net in Fig. 2 is the diagonal matrix given by:

$$D = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \\ t_8 \\ t_9 \\ t_{10} \end{array} \begin{array}{c} \begin{array}{cccccccccc} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & t_8 & t_9 & t_{10} \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & d_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & d_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & d_2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & d_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & d_4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Thus, $(A^-)^T D\ x = (0\ 0\ 0\ (m+1)d_1\ d_2\ 0\ 0\ 0\ d_3\ d_3\ d_4\ d_2+d_4\ 0)^T$, and we find the following to evaluate Equation (4):

$y_1^T(A^-)^T Dx/y_1^T M_0 = 0$

$y_2^T(A^-)^T Dx/y_2^T M_0 = (d_2 + d_4)/r$

$y_3^T(A^-)^T Dx/y_3^T M_0 = d_3/n$

$y_4^T(A^-)^T Dx/y_4^T M_0 = d_3/k$

$y_5^T(A^-)^T Dx/y_5^T M_0 = ((m+1)d_1 + d_2 + d_3 + d_4)/n$

Therefore, the minimum cycle time of firing transitions t_1 and t_2 m+1 times, t_4 m times, and each of the rest of transition once in the net in Fig. 2 is given by

$$Max\{0, (d_2 + d_4)/r, d_3/n, d_3/k, ((m+1)d_1+d_2+ d_3+d_4)/n\}$$

$$= Max\{(d_2 + d_4)/r, d_3/k, ((m+1)d_1+d_2+ d_3+d_4)/n\} \qquad (5)$$

4. Applications of The Model

The minimum cycle time for the timed net in Fig. 2 given by Equation (5) corresponds to the minimum time needed to check if the update is necessary m+1 times, update once and do not update m times among n moving objects.

Example 1: Consider the net in Fig. 1, where we assume that the four time delays are given by the following triangular fuzzy numbers. (An example of a triangular fuzzy number is shown in Fig. 3.)

d_1 = (0.0001, 0.0002, 0.0003) time units, which means that the moving objects check if there is an update generated can be done most likely in time 0.0002 time unit, but possibly between 0.0001 and 0.0003 time unit.

d_2 = (0.005, 0.01, 0.015) time units, which means that the update messages can be sent to the DBMS most likely in 0.01 time unit, but possibly between 0.005 and 0.015 time units.

d_3 = (0.95, 1, 1.05) time units, which means that updating a transaction can be executed most likely in 1 time unit, but possibly between 0.95 and 1.05 time units.

d_4 = (0.005, 0.01, 0.015) time units, which means that the returned information can be sent back to moving objects most likely in 0.01 time unit, but possibly between 0.005 and 0.015 time units.

In addition, suppose that there are n = 1000 moving objects and GPS receivers, r = 10 wireless agents, and k = 1 DBMS processor, and m = 50, the average number of times that update is not needed per each update needed.

From Equation (5), we find the minimum cycle time = d_3/k (the time delay at the DBMS) = (0.95, 1, 1.05) time units in the form of a triangular fuzzy number, which has the possibility distribution shown in Fig. 3.

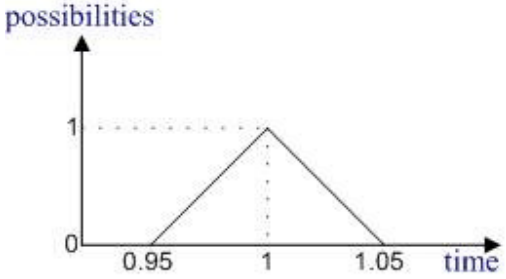

Fig. 3. The possibility distribution of the triangular fuzzy number – the minimum cycle time in Example 1.

That is, the minimum cycle time is most likely 1.0 time unit and can be between 0.95 and 1.05 time units with smaller and smaller possibilities as it deviates from the value 1.0. But there are no possibilities outside of the interval [0.95, 1.05], i.e., it can not be less than 0.95 and more than 1.05 time units.

If the GPS receiver collects the location information every 2 time units, each object can complete one update in (0.95, 1, 1.05) time units, so the system is safe. According to Equation (5), we can increase $(d_2 + d_4)/r$, d_3/k and $((m+1)d_1 + d_2 + d_3 + d_4)/n$ up to d_3/k without affecting the minimum cycle time. For instance, we can decrease the number of wireless agents r to 1, which can reduce the cost paid for wireless communication services. Here we assume that the wireless communication cost depends only on the time of the communication, although some services can depend on the size of messages sent.

Suppose that the GPS receiver collects the location information more frequently, e.g., every 1 time unit. Then the above minimum cycle time (0.95, 1, 1.05) time units for one 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 d_3/k, either by decreasing d_3 or increasing k. The delay d_3 can be reduced by upgrading or improving the DBMS software so as to speed up transactions in the DBMS. The value of k can be increased by adding more DBMS processors. For the DBMS with more than one processors, the database can be processed in parallel.

5. Conclusion and Suggestions for Future Work

This paper has presented a fuzzy-timing high-level Petri net (FTHN) model for updating a MOD. To the best of our knowledge, this is the first time that a Petri net method has applied to the MOD systems. Presented also is a nice application of the minimum cycle time method for estimating the time duration to complete one cycle of update. Once S-invariants are found, the minimum cycle time performance can be computed fast simply by matrix multiplications expressed in Equation (4). The presented method of performance evaluation is *scalable*, since S-invariants and Equation (4) can be computed in a polynomial complexity [6]. Although the FTHN model presented is a general, high-level model for updating a MOD, more complex, and realistic models can be obtained by refining the present model. For example, transitions t_5 and t_8 can be refined in order to model wireless communication protocols. Transition t_3 and t_4 can be extended for a specific update policy. Transition t_6 and t_7 also can be expanded to simulate a specific DBMS architecture, e.g., a distributed DBMS for wireless networks or a centralized DBMS for road networks. As stated in the introduction, there have been proposed many different strategies and policies for updating a MOD. It is interesting and challenging to model these different updating policies using FTHNs and to compare their minimum cycle time performance.

Petri nets are suitable for modelling the concurrency control of concurrent systems. It would be interesting to refine the presented FTHN model to reflect the concurrency control of MOD systems.

References

1. O. Wolfson: Moving objects information management: The database challenge. In the Proceedings of the 5th Workshop on Next Generation Information Technologies and Systems (NGITS'2002), Israel June 2002.
   Available: http://www.cs.uic.edu/~wolfson/html/mobile.html.
2. O. Wolfson, A.P. Sistla, S. Chamberlain, and Y. Yesha: Updating and Querying Databases that Track Mobile Units. Special issue of the Distributed and Parallel Databases Journal on Mobile Data Management and Applications, 7(3), 1999.
3. A. Bar-Noy, I. Kessler, and M. Sidi: Mobile users: To update or not to update? ACM/Baltzer *Wireless Networks Journal*, 1(2), 1995.
4. O. Wolfson and H. Yin: Accuracy and Resource Consumption in Tracking and Location Prediction. In Proceedings of the 8th International Symposium on Spatial and Temporal Databases, Santorini Island, Greece, July 24-27, 2003.
5. T. Murata: Temporal Uncertainty and Fuzzy-Timing High-Level Petri Nets. In Application and Theory of Petri Nets 1996, Lecture Notes in Computer Science, Vol. 1091, Springer, New York, pp. 11-28.
6. T. Murata: Petri Nets: Properties, Analysis and Applications. In Proceedings of the IEEE, Vol. 77, No. 4, April, 1989, pp.541-580.
7. T. Murata, T. Suzuki and S. Shatz: Fuzzy-Timing High-Level Petri Nets (FTHNs) for Time-Critical Systems. In J. Candoso and H. Camargo (editors) "Fuzziness in Petri Nets" Vol. 22 in the series "Studies in Fuzziness and Soft Computing" by Springer Verlag, New York, pp. 88-114, 1999.
8. Y. Zhou, T. Murata, and T. DeFanti: Modeling and Performance Analysis Using Extended Fuzzy-Timing Petri Nets for Networked Virtual Environments. IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics, Vol. 30, No. 5, October 2000, pp.737-756.