International Journal of Computer Theory and Engineering, Vol. 1, No. 1, April 2009
1793-8201

Location Updating Strategies in Moving Object Databases

H. M. Abdul–Kader

***Abstract*—Recent advances in wireless, communication systems have led to important new applications of Moving object databases (MOD). Typical application examples of moving object databases (MOD) might include mobile computing, mobile E-Commerce, Traffic police, taxi dispatchers and weather reporting services. The mobile computing capabilities allow users to manage their work while they are moving. In order to manage such moving objects in database management systems (DBMS) an updating strategy for moving object is required. In this paper the problem of maintaining the current location of moving object in databases and the required updating strategy for moving object will be investigated. An implementation of two updating strategy will be introduced. Those updating strategies are distance and deviation updating polices. This implementation is based on the generation of spatial-temporal data for the moving objects (e.g. cars). The moving objects in the simulated system are moving through a network street. Finally a complete comparison of the predicted moving object path and the actual path is given.**

***Index Terms*—Moving object databases (MOD), location management, Location updating strategies.**

I. Introduction

Recent advances in wireless communication systems and GPS (Global Position system) are the main forces that make position tracking of moving objects are feasible. As a result a widely interest of many new applications that depend on location management can be found in the literature[1].Tourist services, mobile E-commerce and digital battlefield are examples of this applications[1]-[3]. 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 [4], leading to a rather discrete model. DBMS technology provides a foundation for efficiently answering queries about moving objects. However, there is a critical set of capabilities that have to be integrated, adapted, and built on top of existing DBMS's in order to support moving objects databases. These capabilities are added with other things to support spatial and temporal information [4]. Temporal databases [5] and spatial databases [1] were studied for modeling moving object . Fig. 1 shows the relation between moving object database, spatial database, and temporal database. However, the problem is further complicated because the location of a moving object is inherently uncertain, i.e. the position reflected in the database at a particular time will usually not be identical to the actual position of the moving object at that time [1], [3].

The next problem is how to set the uncertainty bounds for a moving object.

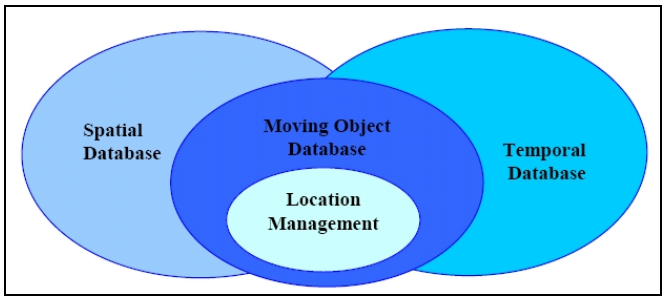

Fig.1 Moving object database, spatial database, and temporal database relations.

So in many MOD applications a continuous model for these dynamic objects will be essential in order to mange such moving objects [4]- [6]. In this case an updating strategy for moving object is required. The objective of this strategy is to accurately maintain the current location of moving object while minimizing the number of update. The most common approach is distance update policy which updates the database every X distance of units. So it provides a certain error in response to a query about the location of any object. The answer is within a circle of radius X centered at location L(which provided in the last updating for database).This is approach is used in many applications due to its simplicity [7], [8]. An alternative approach called a deviation policy which concerned with motion on the road network given by map can be shown in [9]. This approach improves the performance of tracking by location prediction [9]. But it allows for location error as distance policy. The advantage of the deviation policy decreases as the uncertainty threshold increase. The main objective of this paper is to study the trade-off between number of update messages and information accuracy in designing MOD systems using different update policies. This objective will be achieved by minimizing overhead in the communication cost and minimizing penalty in the uncertainty of the moving object location.

This paper is organized as follows first a review for spatio-temporal data Model for MOD is given in section 2. An overview for distance and deviation updating strategies will be presented in section 3. The proposed architecture for the previously discussed updating strategy is introduced in section 4. Simulation results for the proposed architecture

Manuscript received January 26 2009 accepted March 18 2009.

1- H. M. Abdul -Kader is a lecturer in Faculty of Computers and Information, , Minufiya University, Shabin-Elkom, Egypt

DOI: 10.7763/IJCTE.2009.V1.11

will be shown in section 5. Discussion of the obtained results is introduced in section 6. Finally conclusion will be drawn in section 7.

II. Moving Object Spatio-Temporal Data Model

New applications which need frequently update are appeared, so the need to design systems that can deals with continuously changing of data is become essential (e.g. Moving Object Spatio-temporal (MOST) data model [4]).The MOST data model gives a solution to continuously changing location problem. This model represents the location as a function of time changing even without an explicit update. For example, a car location of is given as a function of its motion vector (e.g. north at 50 mile/hour). Although the motion vector of an object can be changed but this change is less frequent than the position change. This model use dynamic without explicitly update. This model considers database as a set of object-classes, where each class has a set of attributes which is either static or dynamic attribute. Some of object classes are spatial which is point class, a line class or polygon-class [5]. Point object classes are mobile or stationary if the object class is stationary its location will be represented by two static attributes Lx and Ly which representing the x and y coordinate of the object. But if the object is mobile, its location is considered a dynamic attribute and represented by seven sub attributes as follow:

1- $L_{xstartposition}$: is x coordinate of object position at start time or x coordinate of last update.

2 - $L_{ystartposition}$: is y coordinate of object position at start time or y coordinate of last update.

3 - L_{route}: is a line spatial object to the route on which object is moving.

4- $L_{starttime}$: is the time when moving object at location $L_{xstartposition}$ and $L_{ystartposition}$ and also the time of the last update assuming that database updates are instantaneous [10].

5- $L_{direction}$: is a binary indicator having value of 0 or 1(these values may correspond to north-south or east-west or the two end points of the route).

6 - S_{peed} *:* linear function of the form $f(t) = b.t$ its defined by speed b of the moving object and gives the current distance from start position or last location update as a function of time t elapsed since last location update.

7- $U_{ncertainty}$: is either a constant or a time function of the number t time units elapsed since $L_{starttime}$. It represents the threshold on the location deviation; when the deviation reaches the threshold the moving objects send a location update message.

The route-distance between two points on a given route can be defined as the distance along the route between those two points. Assuming it is straightforward to compute the route-distance between two points. The database location of a moving object at a given point in time is defined as follows:

- At time $L_{starttime}$ the database Location is the pair of $(L_{xstartposition}, L_{ystartposition})$. While the database location at time $(L_{starttime} + t)$ is the point (x, y) which is at route-distance $(L_{speed} .t)$ from the point *with* coordinates $(L_{xstartposition}, L_{ystartposition})$.

Intuitively, the database location of a moving object m at a given time point t is the location of m as far as the DBMS knows; it is the location that is returned by the DBMS in response to a query entered at time t that retrieves m's location. Such a query also returns the uncertainty at time t, i.e. it returns an answer of the form: m is on L_{route} at most $L_{.uncertainty}$ ahead of or behind $(L_{xstartposition}, L_{ystartposition})$ coordinates. We assume at the beginning of the trip the moving object writes all the sub attributes of the location attribute. Subsequently, the moving object periodically updates its current location and

III. Moving Object Location updates strategies

Tracking moving objects are one of the most common requirements for many location management services. Since, the location of moving object changes continuously but the database location of the moving object cannot be updated continuously; therefore, an updating strategy for moving object is required [4]. The objective of this strategy is to maintain the current location of moving object accurate while minimizing the number of update messages. It is obvious that the more data is updated so, the data is more accurate. However, the cost of updating data increases as more frequently the data is updated. That is, there is the trade-off between number of update messages and information accuracy in designing MOD systems [9]. In the literature there are many updating polices. This section introduces the problems of updating strategies for moving object databases. Followed by, different types of location updating strategies are discussed.

A. The distance-based policy

The distance-based policy is a traditional approach used in all existing commercial transportation systems for the moving object or the cellular network to update periodically the location database; e.g. every *2* miles as shown in Fig. 2. From this figure the moving object *m* starts its trip at location L_0, and after *2* miles sends update message to the database sever to update the previous location to new location (L_1). After the moving object moved *2* miles from L_1, it updates the database server with new location L_3 and so on. The advantage of the distance update policy, which updates the moving object *m* every *x* distance units that is, it provides a bound on the error in response to database queries. Specifically, in response to a query: "*what is the current location of m?"* the answer is within a circle of radius *x*, centered at location *L* (provided in the last database update). Similarly, in context awareness application, if a Personal Digital Assistant (PDA) can hold images or graphics that pertain to an interval of *10* meters, then the moving object will provide its location every *10* meters, and get the next set of images from the database. Therefore, this policy is used in many applications due to its simplicity [7].

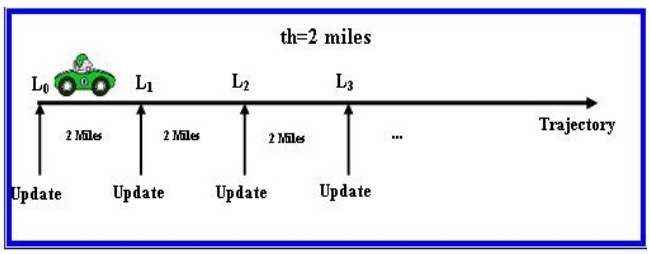

Fig. 2 Distance-based policy

B. *The deviation policy*

The deviation-based policy is another updating strategy that concerned with moving object motion on a road network given by map [7]. Since, the moving object ***m*** updates the server that track its position with all location sub-attributes in the beginning of the trip. The moving objects have various classes (e.g. cars, trucks, and vehicles). Each class has a specific speed. When the moving object in a specific class with *V.x* , *V.y*, *L.x*, *L.y* update the database at time *L.starttime*, sub-attributes the database location (*L.x, L.y*) can be expected at any time *t*. Since, the moving object does not travel at exactly speed (*V.x, V.y*) so, the actual location of the moving object deviates from its expected location. When the deviation reach to a given *threshold* the moving object will update the database with the actual location. Thus in response to a query entered at time *t* (*L.starttime* + *t*) that retrieve *m* location, the *DBMS* returns the location of *m* within a circle with radius equals the *threshold* value centered at a point of the last location update.

Note that the above deviation-based policy allows also a maximum location error equal threshold and the advantage of the deviation policy decreases as the threshold increase. To compare the deviation update policy with traditional popular distance update policy (periodically updates the database every x distance units) the deviation policy is more efficient than the distance policy in terms of number of update required by each policy given the same threshold. Since, the objective is not only, to make each update more efficient but also reducing the number of required update. In distance update, policy the database cannot predict the next location but the database location does not change until the next location update, occurs in contrast the deviation policies can expect the next location as the moving object move in the same street [7], [14].

Other literature of road networks, [2], [8], [9] propose a set of dead-reckoning policies when the route and the destination of the trip are 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 [7] 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.

C. *Network-based generation of spatio-temporal data*

Previous approaches for generating spatio-temporal data do not consider that moving objects often follow a given network [3], [5]. Therefore, benchmarks require datasets consisting of such "network-based" moving objects [5]. In this section, the most important properties of network-based moving objects are presented and discussed. Essential aspects are the maximum speed and the maximum capacity of connections. These characteristics are the basis for the specification and development of a new generator for spatio-temporal data. This generator combines real data (the network) with user-defined properties of the resulting dataset. In this paper, a framework is proposed where the user can control the behavior of the generator by re-defining the functionality of selected object classes. The following requirements are needed for generating spatio-temporal data:

One important technique to evaluate spatial and spatiotemporal database systems and their components are experiments. Many parameters can be measured investigating, e.g., an access method [8]. The most important parameters are the query time (I/O and CPU) for essential queries, the time for building up or modifying the index, and the space requirements. Furthermore, the data used by the experiments are of high importance. We can distinguish between synthetic data generated according to statistical distributions and real data, which originate from real-world applications. However, it is difficult to assess the performance of real applications by employing synthetic data. The use of real data tries to solve this problem. In this case, the selection of the data is essential. For non-experts it is often difficult to decide whether the data reflect a "realistic" situation or not.

IV. The Proposed System Architecture

In this section the proposed architecture and the simulation results are presented. The proposed architecture is shown in Fig. 3. This model consists of many main subsystems.

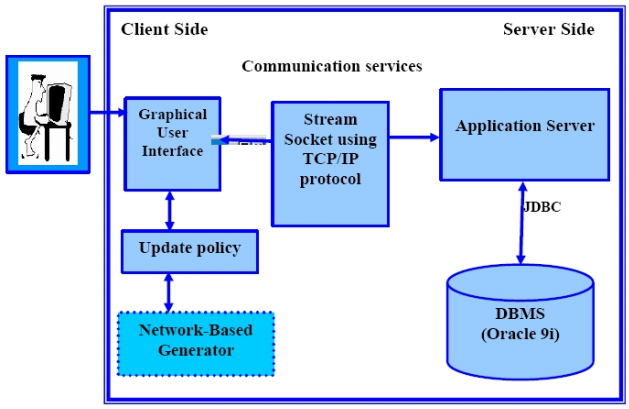

Fig. 3 the framework of the proposed model based on Network-Based Generator.

The main subsystem is the moving object subsystem which normally contains a GPS system and local database. Where the current position of moving object is stored and updated by a GPS at a fixed rate (e.g. 2 sec.). The local database is managed by a DBMS which supports triggers. A trigger fires and updates the central database when the deviation bound is reached. Through a wireless communication network the moving object connected to database server. Where the location update is send to database server to update the position attributes of the database. The database server subsystem is used maintain the current location of large number of moving object, and also to query a set of real moving objects. Location data is generated by each moving object. The moving object subsystem updates the Central Database when deviation bound is reached. The query processor is used to query the central database. The pseudo code of the distance update the

deviation update policies can be described as shown in figures 4 and 5 respectively.

V. Simulation Results

The first part of simulation considers a comparison between different values for the threshold value to select the suitable value to complete the required comparison between the selected updating strategies. At the beginning of simulation the moving object update its local database and the central database at the server with its location sub attributes (x, y, v_x , v_y , t , L_{streat}) and with threshold value(th). Also assuming that every (t= 2 second) for example both position and speed supposed to be known as it coming from the GPS. These values inserted in the local database of the moving object. When these values are inserted fire a trigger which computes

Input: *Database of moving objects contains the locations of moving objects with times and routes.*

Output: *Locations of moving objects, the route length, the total number of updates during the trip, and number of Update Per unit Length (UPL)*

1. i=0, Total_no_updates=0, Max_time_satmp=15
2. **Do**
   - **Read** the current location ($L.x_i$, $L.y_i$, t_i) and previous location ($L.x_{i-1}$, $L.y_{i-1}$, t_{i-1}) of the moving object .
   - **If** the Euclidian distance $\sqrt{(L.x_i - L.x_{i-1})^2 + (L.y_i - L.y_{i-1})^2}$ > *threshold.*
   - **Then**

     Update the database with *Current location* ($L.x_{i-1}$, $L.y_{i-1}$, t_{i-1}) = *Current location* ($L.x_i$, $L.y_i$, t_i) , i= i+1,

     Total_no_update= Total_no_update +1.
   - **Else**

     Current location ($L.x_i$, $L.y_i$, t_i) = *Previous location* ($L.x_{i-1}$, $L.y_{i-1}$, t_{i-1}), i= i+1.
3. **While** ($i \leq Max\_time\_stamp$).
4. The route length = diff (start point, destination point) in x, y directions.
5. **UPL**=Total_no_update_messages /the route length.

Fig. 4 The pseudocode of the distance update policy

the expected values (x_exp , y_exp)of the position. Since x_exp =x + v_x .t where x is the x coordinate at the last update, t is the time elapsed from last update and v_x is the speed of moving object at last update. In the same y_exp = y + v_y.t. Then compute the deviation value which represent difference between the position of moving object coming from the GPS (actual position) and these values computed by the database trigger (the expected position). If the value of the difference is greater than the value of (th) the local database of the moving object is updated with the actual position and actual speed. Also update the central database with the actual position and speed. Otherwise if the difference value is less than the (th) no update occur. In our implementation we assume the moving object move for 20 second every two second the position and speed (x, y, v_x, v_y) are known and we compute the expected value of the position based on the given sub attributes. Then compute the deviation and if the deviation >= a specific threshold (given in $L_{uncertainty}$ sub attribute) an update occur. In Fig. 6 that represent the actual and expected path through 20 second at th = .01 mile. We note that central database need to be update with actual location only three times at point a, b and c. Since the deviation is greater than the value of th instead of updating the database every two second. In Fig. 7 we reduce the value of threshold (th=.001) we observe that the number of update increase but the uncertainty decrease.

Input: *Database of moving objects contains the locations of moving objects with times and routes.*

Output: *Locations of moving objects, the route length, the total number of updates during the trip, and number of Update Per unit Length (UPL)*

1. i=0, Total_no_updates=0, Max_time_satmp=15, speed
2. **Do**
   - **Read** the current location ($L.x_i$, $L.y_i$, t_i), previous location ($L.x_{i-1}$, $L.y_{i-1}$, t_{i-1}), and speed of the moving object .
   - **Calculate** the expected location($L.x_{Expi}$, $L.y_{Expi}$)
   - **If** the Deviation distance $\sqrt{(L.x_i - L.x_{Exp_i})^2 + (L.y_i - L.y_{Exp_i})^2}$ > *threshold.*
   - **Then**

     Update the database with *Current location* ($L.x_{i-1}$, $L.y_{i-1}$, t_{i-1}) = *Current location* ($L.x_i$, $L.y_i$, t_i) , i=i+1,

     Total_no_update= Total_no_update +1.
   - **Else**

     Current location ($L.x_i$, $L.y_i$, t_i) = *Previous location* ($L.x_{i-1}$, $L.y_{i-1}$, t_{i-1}), i= i+1.
3. **While** ($i \leq Max\_time\_satmp$).
4. The route length = diff (start point, destination point) in x, y directions.
5. **UPL**=Total_no_update_messages /the route length.

Fig. 5 The pseudocode of the deviation update policy.

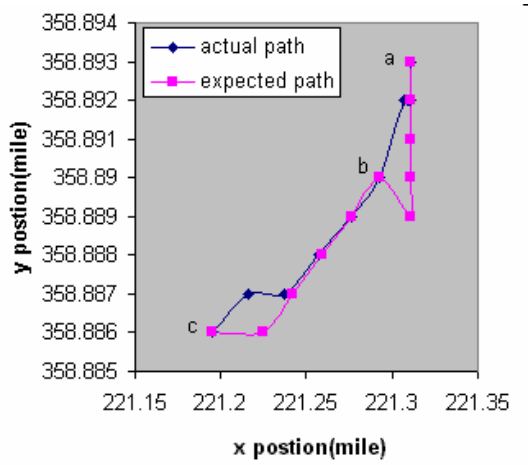

Fig. 6 The actual and expected path at (th =.01)

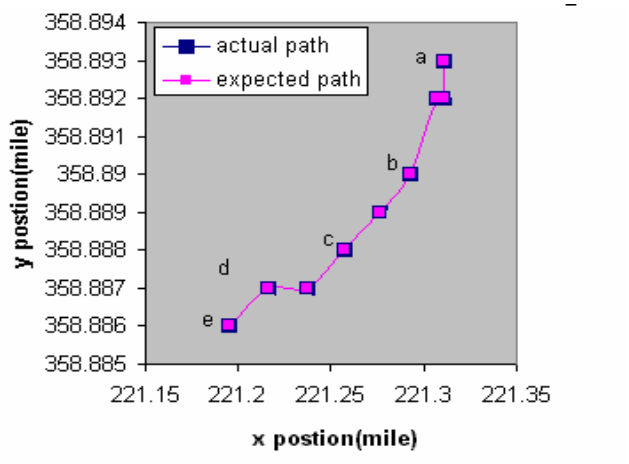

Fig.7 The actual and expected path at (th =.001)

To complete the evaluation of the proposed data model the value of *UPL* of this moving object should be at the threshold value equals *0.55*. In deviation update policy, the total number of update messages equals *3* while the total length of routes equals 16.617 *km*. By substituting in UPL= *total nu. of update messages/total length of routes* then *UPL= 3/16.617= 0.181*. However, in the distance update policy the value of *UPL=16/16.617=0.963*. The number of update messages per unit length with threshold value ranges from *0.05* to *2.2* is shown in Fig. 8.

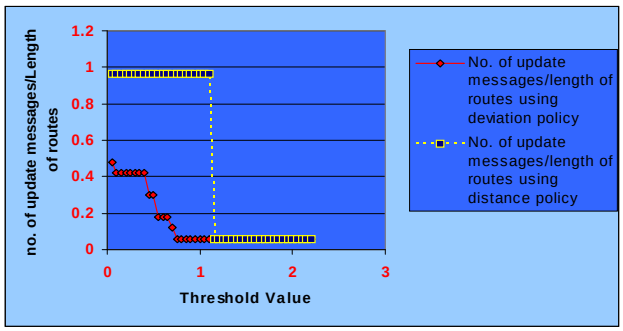

Fig. 8 Number of update messages per unit length using distance and deviation update policy.

The average number of update messages per unit length with threshold value ranges from 0.05 to 2.2 using the distance update policy and the deviation update policy is shown in Fig. 9.

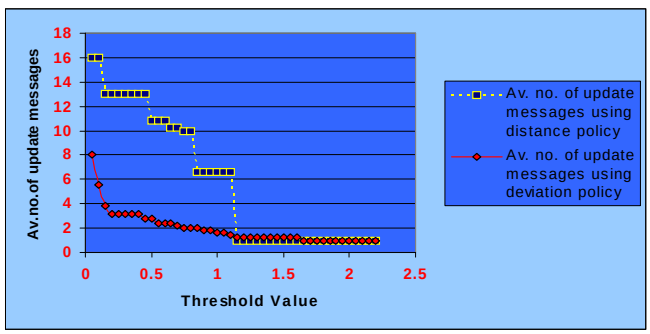

Fig. 9. Average number of update messages with threshold values.

From this figure, the average number of update messages (equals 2.4) using deviation update policy is smaller than the average number of update messages (equals 10.8) using the distance update policy at the same threshold value (equals 0.55). In additional to, when the threshold value equals 1.65 the average number of update messages in the distance and the deviation policy is an identical. The average number of update messages per unit length of routes with threshold value ranges from 0.05 to 2.2 using the distance update policy and the deviation update policy is shown in Fig. 10.

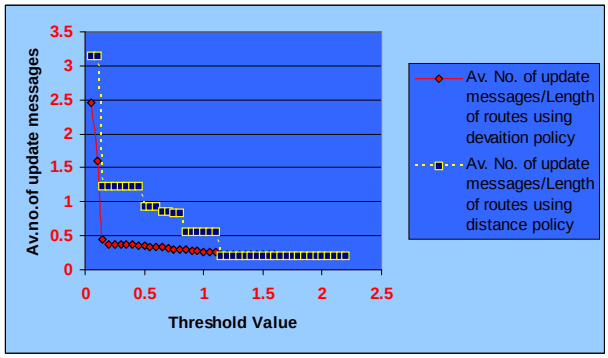

Fig. 10 Average number of update messages per unit length of routes with threshold values.

From this figure, the average number of update messages per unit length of routes (equals 0.3308) using the deviation update policy is smaller than the average number of update messages per unit length of routes (equals 0.9266) using the distance update policy at the same threshold value (equals 0.55). In additional to, when the threshold value equals 1.65 the average number of update messages per unit length of routes in the distance and the deviation policy is an identical. Table I shows the comparison between the deviation update policy and the distance update policy [10].

TABLE I: COMPARISONS BETWEEN THE DEVIATION AND THE DISTANCE UPDATE POLICY

	Deviation update policy	**Distance update policy**
Threshold Value	Fixed for all location update messages. **Th=0.55**	Fixed for all location update messages. **Th=0.55**
No. of update messages	Require less number of update messages (**3**).	Require more number of update messages (**16**).
Location Prediction	Can expect future location so, it is respond to future query.	Cannot expect the future location.
Information cost	**39.8** units of money	**104.8**
UPL	**0.181**	**0.963**

VI. RESULTS DISCUSSION

The main results of the data model can be summarized as follows:

The model uses the network-based generator to get the appropriate updating policy to model moving objects

database. This data model is consists of three parts, the first part consists of graphical user interface, update policy and network –based generator. This part is responsible for enabling the user to redefine the required function, generating realistic database and maintaining the location of moving objects with minimum number of updates. The part, used to help the moving objects to connect with database server and vice verse. The third part, consists of application sever, JDBC, and database management system (Oracle 9i). This part is responsible for helping the user to access the database through the interface between the user and database system.

- This model uses the information cost to compare between updating policies (the deviation update policy, the distance update policy). In additional to, this model presents a new method to distinguish between the updating policies. This method is called the number of update message per unit length (UPL).

From the simulation results of the data model presented in section V , the information cost (equals 39.8) of deviation update policy is smaller than information cost (equals 104.8) of the distance update policy. In additional to, the number of update messages per unit length (UPL= 0.181) in deviation update policy is smaller than the number of update messages per unit length (UPL= 0.963) in distance update policy. Furthermore, number of update messages (equals 3) in deviation update policy is smaller than the number of update messages (equals 16) in distance update policy at the same threshold (th=0.55). Therefore, the deviation update policy is better than distance update policy (theses comparisons are shown in Table I). Fig. 10 shows the average number of update messages per unit length of routes using the distance and the deviation update policy.

VII. Conclusions

In this paper we addressed tracking of moving objects, i.e. the continuous representation of their location in the database. A framework of the proposed model based on Network-Based Generator is introduced. Two updating strategies are implemented based on the proposed model architecture namely; distance and deviation updating policies. Simulation results show the number of updating triggers depends up on the adjusted threshold value for both updating strategies. Finally we can conclude from simulation results that the deviation update policy is better than the distance update policy based on the information cost and the number of update messages per unit length.

References

[1] Hartmut, R., and Schneider, M. "Moving Object Databases", Morgan Kaufmann Publishers 2005.
[2] Ralf Hartmut Güting, Markus Schneider: Moving Objects Databases, web site: http://www.informatik.fernuni-hagen.de/import/pi4/Lehre/Kurse/1676-%23mod/KE1.pdf.
[3] Bin Lin · Jianwen Su "OneWay Distance: For Shape Based Similarity Search of Moving Object Trajectories " Geoinformatica 12:117–142 DOI 10.1007/s10707-007-0027-y-(2008)
[4] Ouri Wolfson, Liqin Jiang, , Sam Chamberlain, Bo Xu : Moving Object Databases : Issues and Solutions, SSDBM 1998, pages 111-122, 1998.
[5] Ouri Wolfson, Liqin Jiang, A.Prasad Sistla, Sam Chamberlain, Naphtali Rishe, and Minglin Deng: Databases for Tracking Mobile Units in Real Time, Proceedings of the Seventh International Conference on Database Theory (ICDT), pages 169-186, 1998.
[6] Ouri Wolsofon, Prasad Sistla, Bo Xu, Jutai Zhou, Sam Chamberlain, Yelena Yesha, and Naphtali Rishe : Tracking Moving objects using database technology in DOMINO, proceedings of NGITS 99, the fourth workshop on next generation information technologies and systems, pages 112-199, July 1999.
[7] QUALCOMM Inc.: http://www.qualcomm.com
[8] At Road Inc.: http://www.atroad.com/
[9] Ouri Wolfson and Huabei Yin: Accuracy and resource consumption in tracking and location prediction, proceedings of 8 th international symposium on spatial and temporal databases, pages 325-343, July 2003.
[10] R.Snodgrass and I.Ahn: The Temporal Database, IEEE computer, Sept. 1986.
[11] Brinkhoff Thomas, Generating Traffic Data. IEEE, 2003.
[12] Brinkhoff Thomas, A Framework for Generating Network-Based Moving Objects. Tech.Report of the IAPG, http://www2.fh-wilhelmshaven.de/oow/institute /iapg /personen /brinkhoff/paper/TBGenerator.pdf
[13] Y. Theodoridos and M. Nascimento. Generating Spatiotemporal Datasets on the WWW. ACM SIGMOD Record, pages 29(3):39-43, 2000.
[14] Y. Theodoridis, J.R.O. Silva, and M.A. Nascimento. On the Generation of Spatiotemporal Datasets, in Proceedings 6th International Symposium on Large Spatial Databases, Hong Kong, China, Lecture Notes in Computer Science, pages 1651:147-164, 1999.

Dr. H. M. Abdul-kader obtained his B.S. and M.SC. (by research) both in Electrical Engineering from the Alexandria University , Faculty of Engineering , Egypt in 1990 and 1995 respectively. He obtained his Ph.D. 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 Information systems department, Faculty of Computers and Information, Minufiya University, Egypt since 2004. He has worked on a number of research topics and consulted for a number of organizations. He has contributed more than 30+ technical papers in the areas of neural networks, Database applications, Information security and Internet applications.