Managing and Querying Moving Objects in Networks

Jae-Chul Kim
LBS Research Team,Telematics Research Division, ETRI
161 Kajong-Dong, Yusong-Gu, Taejon, 305-350, Republic of Korea
kimjc@etri.re.kr

Tae-Wook Heo
Tae-Wook HeoLBS Research Team,Telematics Research Division, ETRI
161 Kajong-Dong, Yusong-Gu, Taejon, 305-350, Republic of Korea
htw398@etri.re.kr

Jai-Ho Lee
Tae-Wook HeoLBS Research Team,Telematics Research Division, ETRI
161 Kajong-Dong, Yusong-Gu, Taejon, 305-350, Republic of Korea
snoopy@etri.re.kr

Kwang-Soo Kim
LBS Research Team,Telematics Research Division, ETRI
161 Kajong-Dong, Yusong-Gu, Taejon, 305-350, Republic of Korea
enoch@etri.re.kr

Abstract: We model a moving object as a sizable physical entity equipped with GPS, wireless communication capability, and a computer such as a PDA and mobile phone. Furthermore, we have observed that a real trajectory of a moving object is the result of interactions among moving objects in the system yielding Multi-points instead of a line segment. In this paper, the new types and operations are integrated seamlessly into the moving object framework to achieve a relatively simple, consistent and powerful overall model and query language for constrained and unconstrained movement.

Keywords: Moving Object, GPS, Location Based System.

1. Introduction

Some of the interest is spurred by current trends in consumer electronics. Wireless networking enabled and position-aware (i.e. GPS equipped) devices such as PDAs, on-board units in vehicles, or even mobile phones have become relatively cheap and are predicted to be in widespread use in the near future. This will lead to many new kinds of applications such as location-based services. At the same time a huge volume of movement information (sometimes called trajectories) will become available and need to be managed and analyzed in database systems.

Moving Objects Databases have become an important research issue in recent years. For modeling and querying moving objects, there exists a comprehensive framework of abstract data types to describe objects moving freely in the 2D plane, providing data types such as moving point or moving region. However, in many applications people or vehicles move along transportation networks. It makes a lot of sense to model the network explicitly and to describe movements relative to the network rather than unconstrained space, because then it is much easier to formulate in queries relationships between moving objects and the network. Moreover, such models can be better supported in indexing and query processing.

2. Data Modeling

1) Modeling Networks

A first idea, which may seem obvious at least to a computer engineer, is to model a transportation network as a directed graph G = (V, E), with set of nodes V and set of edges $E \subseteq V \times V$. The set of possible positions within the network could then be defined as

$$POS(G) = V \cup (E \times (0, 1))$$

That is, an object is either in a node or on an edge; a value from the open interval (0, 1) indicates a relative position on the edge. This view of a network is used for example, in [1,2]. However, there are some indications that this model is not the best one. In the literature, Jensen et al. [3] emphasize that real world networks are quite complex and that a realistic modeling needs in fact multiple representations. One of the important views is the kilometer-post representation which describes positions in the network relative to distance markers on the roads. This is closely related to the concept of linear referencing widely used in the GIS-T literature (geographic information systems in transportation), see for example [4], where again positions are described relative to the length of a road. Linear referencing is also already available in commercial database products such as Oracle Spatial [5].

Defining a data type means to introduce a name for it and to define the set of possible values, i.e., the *domain* or the *carrier set*. We use the algebraic terminology and define carrier sets; for a type *t* its carrier set is denoted as D_t. For the type *network*, the carrier set is:

$$D_{network} = Network \quad (1)$$

The data types *gpoint* obviously depends on existing networks. Let $N = \{N_1, ..., N_k\}$ be the set of networks present in the database. We define types *gpoint* and *gline* with carrier sets:

$$D_{gpoint} = \{(i, gp) \mid 1 \leq i \leq k \wedge gp \in (Loc(N_i) \cup \{\perp\})\} \quad (2)$$

So a value of such a type consists of a network number together with a position or region within that network. The network position may be undefined, which is represented by $\perp$. A network region is a set (of route intervals) which may be empty anyway.

3. System Architecture

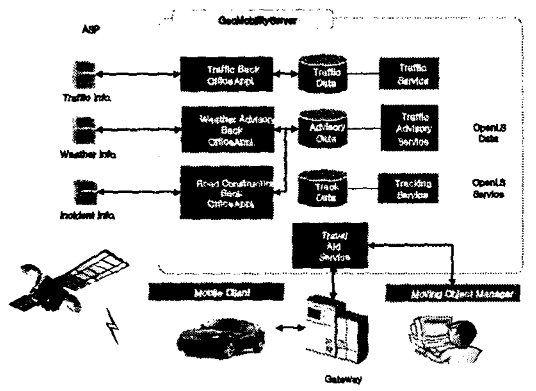

Fig. 1. System Architecture.

Figure 1 shows how the concept GeoMobility Server relates to the other elements of a LBS architecture. The GeoMobility server is an element offering basic functions on which location-based applications are built (the OpenLS Core Services). This server uses open interfaces to access network location capacity (provided through a GMLC, for instance) and provides a set of interfaces allowing applications hosted on this server, or on another server, to access the OpenLS Core Services. The GeoMobility Server also provides content such as maps, routes, addresses, points of interest, traffic, etc. It can also access other local content databases via the Internet.

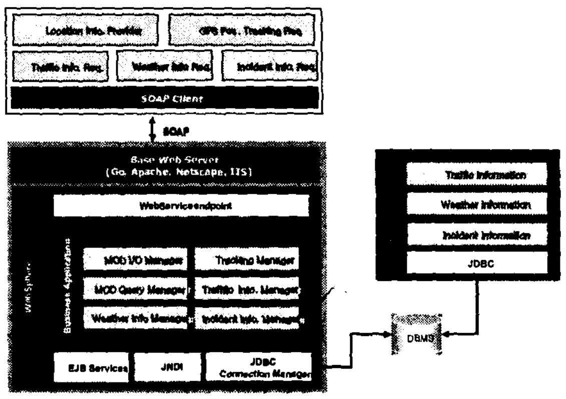

Fig. 2. *Component Architecture.*

As shown in Figure 4, LBS common core components are connected to Geo-Mobility Server. And the interface of core service is implemented with Web-Service. The proposed system consists of Web-Service server and Web-Service client developed from EJB (Enterprise Java Beans). A developing tool is WSAD (Websphere Studio Application Developer) 5.0 and WAS (Websphere Application Server) 5.0 is used in web server. The interface of service platform considering interoperability is implemented on the basis of OpenLS (OGC, 2002a, 2002b, 2002c, draft). Each external CP (Content Provider) gateway communicate with LBS platform to access LBS common core components, also it communicate with its client by using the Servlet.

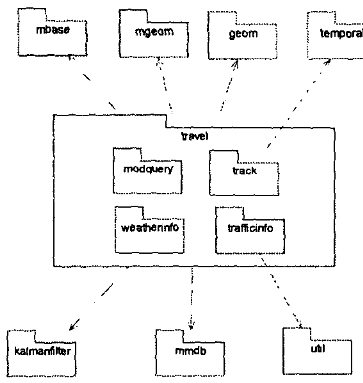

Fig. 3. Package Diagram.

Table 1. Package Discription.

Package Name	Description
travel	Class package for Travel Advisory implementation
modquery	Class package for MO querying
track	MO I/O and map matching class package
weatherinfo	Weather information querying class package
mbase	Class packge for MODB Mbase data processing
mgeom	Class packge for MODB MGeometrydata processing
geom	Class packge for MODB Geometry processing
temporal	Class packge for MODB Temporal processing
kalmanfilter	Kalman Filter algorithm class package for MO Location correction

mmdb	Class package for Connecting Pool managing
util	Utility class package

4. Managing Moving Object

1) Deviation Cost

The deviation has a cost (or penalty) because it can result in incorrect decision making. Observe first that the cost of the deviation depends both, on the size of the deviation and on the length of time for which it persists. It depends on the size of the deviation since decision-making is clearly affected by it. To see that it depends on the length of time for which the deviation persists, suppose that there is one query that retrieves the location of a moving object per time unit. Then, if the deviation persists for two time units its cost will be twice the cost of the deviation that persists for a single time unit; the reason is that two queries (instead of one) will pay the deviation penalty.

2) Updata Cost

The update cost is a nonnegative number representing the cost of a location-update message sent from the moving object to the database. The update cost may differ from one moving object to another, and it may vary even for a single moving object during a trip, due for example, to changing availability of bandwidth. The update cost must be given in the same units as the deviation cost. In particular, if the update cost X is it means the ratio between the update cost and the cost of a unit of deviation per unit of time.

3) Uncertainty Cost

The uncertainty has a cost (or penalty) because a higher uncertainty conveys less information when answering a query. Observe that, as for the deviation, the cost of the uncertainty depends both, on the size of the uncertainty and on the length of time for which it persists.

We use *Kalman filter algorithm*[6] in adjusting uncertain moving object. process state at some time and then obtains feedback in the form of (noisy) measurements. As such, the equations for the *Kalman filter* fall into two groups: time update equations and measurement update equations. The time update equations are responsible for projecting forward (in time) the current state and error covariance estimates to obtain the a priori estimates for the next time step. The measurement update equations are responsible for the feedback-i.e. for incorporating a new measurement into the a priori estimate to obtain an improved a posteriori estimate.

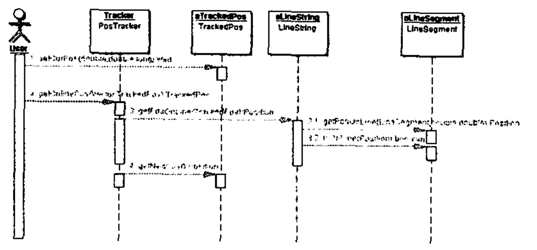

Fig. 4. Sequence Diagram of adjusting uncertain moving object.

5. Querying Moving Object

1)Time-Series View

Moving Point Objects have time-series characteristic. Therefore it support Time-Series function such as Table 2.

A characteristic of Time-Series function is that target of operation is not location information of moving points at any time operated by means of moving object function but real location data stored in database. In other words, The first() function returns k location information from first value of moving point, and last() function returns k location information from last value of it. The after() function returns similarly the nearest location information after time t in parameter, and The before function returns the nearest location information before time t in parameter.

Table 2. Time-Series functions of moving points

Classification	Time-Series Operations	
GetTime	$validtime(e_i) = t_i$	$index(e_i) = i$
GetFirstLast	$first(s,k) = e_k$	$last(s,k) = e_k$
GetElement	$next(e_i) = e_{i+1}$ $after(s,t_i) = e_i$	$previous(e_i) = e_{i-1}$ $before(s,t_i) = e_i$
Slice	$slice\_squence(s,k_{from},k_{to}) = s'$ $slice\_sequence(s,t_{from},t_{to}) = s'$	

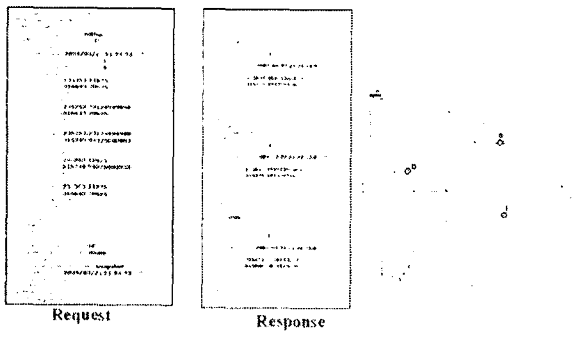

Fig. 5. Querying Moving Object(snap shot).

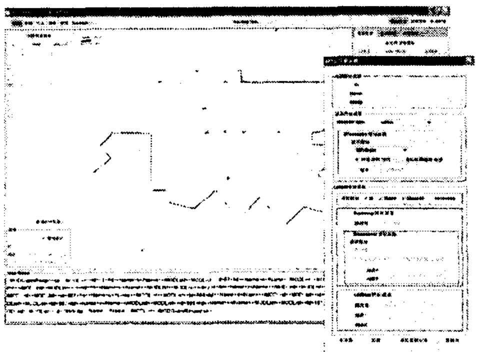

Fig. 6. Querying Moving Object(track).

2)Continuous Moving Objects View

For continuous moving objects view, we propose three kinds of operators, snapshot, slice, and project as shown in Table 3. Snapshot_ prefix operators return a snapshot by the specific dimension described in parameter. These consist of *snapshot_validtime* and *snapshot_value* operator. *Snapshot_validtime* operator return snapshot value object in specific time point. Oppositely, *snapshot_value* return snapshot times by value dimension. Slice operators return sliced moving objects by the specific dimension described in parameters. *Slice_validtime* operator returns a moving object between specific time period. Oppositely, *slice_value* operator returns moving objects sliced value objects. Project operators return value objects projected by another dimension, time or value.

Slice operators are distinguished from snapshot operators in returned values. The former return moving objects, but the latter return snapshot objects in value dimension.

Table 3. Function of moving object model

Classification	Continuous Moving Objects Operations
Snapshot	$snapshot\_validtime(m,t_i) = v_i$ $snapshot\_value(m,c_i,x) \subseteq domain(Time)$
Slice	$slice\_validtime(m,t_{from},t_{to}) = m'$ $slice\_value(m,c_i,x) = m'$
Project	$project\_validtime(m) \subseteq domain(Time)$ $project\_value(m,c_i) \subseteq domain(T_i)$

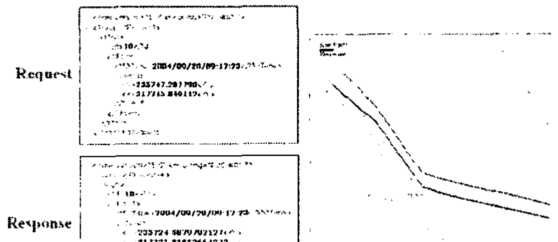

Fig. 7. Tracking Moving Object.

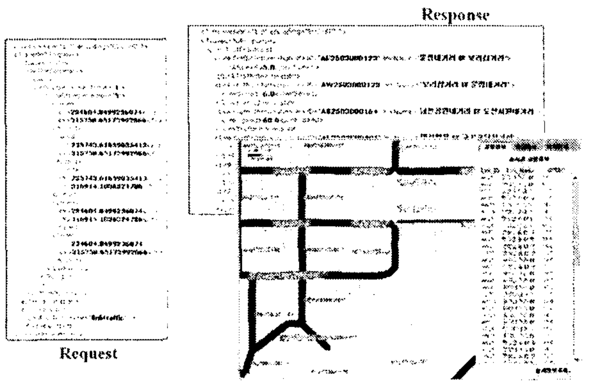

Fig. 8. Traffic Information on Road Networks.

6. Conclusions

We conducted numerous simulations to compare the information cost of location update policies. The parameters of the simulation are the following: the update-unit cost, namely the cost of a location-update message, the uncertainty-unit cost, the deviation-unit cost, and a speed curve, namely a function that for a period of time gives the speed of the moving object at any point in time. We built a simulation testbed which enables us to compare the policies in terms of number of messages, deviation, and uncertainty.

References

[1] R. Frentzos, Indexing Moving Objects on Fixed Networks. *In Proc. of the 8th Intl. Symp. on Spatial nd Temporal Databases*, 2003, 289-305.

[2] T. Brinkhoff, A Framework for Generating Network-Based Moving Objects. *GeoInformatica*: 153-180, 2002.

[3] C. S. Jensen, T.B. Pedersen, L. Speicys, and I. Timko, Data Modeling for Mobile Services in the Real World. *In Proc. of the 8th Intl. Symp. on Spatial and Temporal Databases*, 2003, 1-9.

[4] P. Scarponcini, Generalized Model for Linear Referencing in Transportation. *GeoInformatica* :35-55, 2002.

[5] Oracle Spatial Linear Referencing System User's Guide, Release 8.1.6., Oracle Press, 2000.

[6] Brown, R. G. and P. Y. C. Hwang. 1992. Introduction to Random Signals and Applied Kalman Filtering, Second Edition, John Wiley & Sons, Inc.