위치 인식 컴퓨팅에서 효율적인 위치 관리 기법
(An Efficient Location Management Scheme in Location-Aware Computing)

송 문 배 † 강 상 원 †† 박 광 진 † 황 종 선 †††
(MoonBae Song) (SangWon Kang) (KwangJin Park) (Chong-Sun Hwang)

요 약 이동 객체의 위치를 효율적으로 추적하는 방법을 찾는 일은 위치 인식 컴퓨팅에서 가장 중요한 문제 가운데 하나로 꼽을 수 있다. 이를 위해서 저장된 위치 정보를 효율적으로 갱신하는 프로토콜이 필요하다. 이러한 위치 갱신 전략의 성능은 가정한 이동 패턴에 따라 크게 좌우되는데, 지금까지의 연구는 이동성 문제를 무시하였거나, 시간에 대한 지나치게 단순한 선형함수로만 다루고 있다. 이 논문은 이동 패턴이 복잡한 객체들의 위치 정보를 표현하고 갱신하기 위한 상태 기반 이동 모델을 제안한다. 그리고, 이 모델을 바탕으로 이동성을 인식하는 상태기반 위치 갱신 프로토콜을 제안한다. 성능을 비교하여, 이 방법이 위치 갱신 비용과 대역폭 소모를 두드러지게 줄일 수 있음을 보여준다.

키워드 : 이동 객체 데이타베이스, 위치 관리, 이동 패턴, 위치 갱신 프로토콜, SLUP, SMM

Abstract One of the most important issue in location-aware computing is tracking moving objects efficiently. To this end, an efficient protocol which updates location information in a location server is highly needed. In fact, the performance of a location update strategy highly depends on the assumed mobility pattern. In most existing works, however, the mobility issue has been disregarded and too simplified as linear function of time. In this paper, we propose a new mobility model, called *state-based mobility model* (SMM) to provide more generalized framework for both describing the mobility and updating location information of complexly moving objects. We also introduce the *state-based location update protocol* (SLUP) based on this mobility model. Using experimental comparison, we illustrate that the proposed technique is many times better in reducing location update cost and the communication bandwidth consumption.

Key words : moving objects database, location management, mobility patterns, location update protocol, SLUP, SMM

1. Introduction

Recently, the evolution of mobile computing, location sensing, and wireless networking has created a new class of computing: *location-aware computing*[1]. A location-aware system is one that knows where each user is located and can use the location-specific information anywhere and anytime. In location-aware computing environments, the mobility of mobile client (MC) is emerging in many forms and applications such as database, network and so on. MCs can dynamically change their locations over time. The objects which continuously change their location and extent are called *moving objects*[2-4]. Thus, one of the most important issues in location-aware computing is how to model the location and the movement of moving objects efficiently. Therefore, a software infrastructure for providing location-aware computing environments, called *moving objects database* (MOD), is significantly needed.

In recent times, there is a lot of work on the representation and management of moving objects

† 비 회 원 : 고려대학교 컴퓨터과학기술연구소 연구원
mbsong@disys.korea.ac.kr
kjpark@disys.korea.ac.kr
†† 비 회 원 : 고려대학교 컴퓨터학과
swkang@disys.korea.ac.kr
††† 종신회원 : 고려대학교 컴퓨터학과 교수
hwang@disys.korea.ac.kr
논문접수 : 2004년 1월 13일
심사완료 : 2004년 7월 7일

[2-7]. Wolfson *et al.* present the well-known data model called Moving Object Spatio-Temporal (MOST) for representing moving objects[3]. In the MOST model, the location of moving objects is simply given as linear function of time, which is specified by two parameters: the position and velocity vector for the object. Thus, without frequent update message, the location server can compute the location of a moving object at given time t by linear interpolation: $y(t)=y_0+\overline{v}(t-t_0)$ at time $t>t_0$. The update message is only issued when the parameter of linear function, e.g. $\overline{v}$, changed. In general, we said this update approach *dead-reckoning*. It can provide a great performance benefit in linear mobility patterns. But the performance is decreased by increasing the randomness of mobility pattern. So this approach suffers great performance degradation in non-linear mobility patterns.

In this paper, we look at the mobility model for MOD and an appropriate location update protocol. The purpose of our scheme is to model the overall movement patterns in probabilistic manner. Depending on the temporal locality of mobility patterns, the proposed scheme can greatly reduce the number of update messages.

The rest of this paper is structured as follows: In Section 2, we introduce the characteristics of mobility patterns of real-life objects. The proposed mobility model, called *state-based mobility model* (SMM), will be described in Section 3. In Section 4, we present a new location update protocol called *state-based location update protocol* (SLUP) considering mobility patterns on a per-user basis. Extensive performance evaluation and comparison of proposed scheme with traditional update strategies are also included in Section 5. Finally, the summary and future work are presented in Section 6.

2. Motivation

Consider a traveling salesman who travels several cities for selling commodities. He starts from his company, and moves through an expressway. When he reaches its destination, he strolls around the city selling commodities, then finds a new destination. We anticipate that this model will be able to capture a large part of real-life objects' movements. And it would include the essential elements of mobility patterns such as linear movement, random movement, and stationary state. Whereas, existing mobility models have not express the realistic movements of real-life objects. Thus, it is inevitable that the update cost of a moving object and the average error of accuracy will be increased. In our work, we will classify the whole trajectory of a user into 'pause', 'linear movement', and 'random movement' in the rough(see Figure 1).

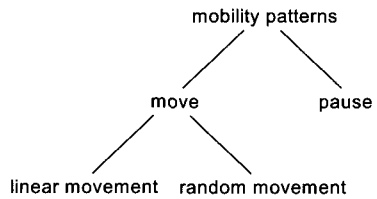

Figure 1 The classification of mobility patterns of reallife objects

As we already know, a great diversity of mobility patterns of real-life objects is quite natural. But, there are some specific repeated patterns in the movements. For example, in the linear movements, the trajectory of an object is almost a line in d-dimensional space. Whereas, if we can't find the implicit knowledge of a specific pattern, let us identify the portion of trajectory as *random walk* or *Brownian movement*. And, of course, we have to consider the temporal pattern of movements as Markovian process. Our approach to the problem of mobility modeling is primarily motivated by the following observations[4,5,6,8].

- A mobile subscriber will mainly switch between two states: *stop* and *move*. A traveling salesman has a tendency to remain in the same state rather than switching states[8].
- The majority of objects in the real world do not move according to statistical parameters but, rather, move intentionally[6,8].
- Moving objects belong to a class. This class restricts the maximum speed of the object. Different groups of moving objects exhibit different kinds of behavior[5,6].

- Motion has a random part and a regular part, and the regular part has a periodic pattern[4].

The above properties are distilled from dozens of papers in both Personal Communication System (PCS) and MOD. Despite of all these observations, the most of existing works have been disregarded and too simplified as linear function of time. Mobility models and its applications are widely studied in location management of PCS environments. Many existing location management proposals use some version of a random mobility model, typically one-dimensional [9]. Modeling the random-walk as a mathematical formula is a simple process without difficulty. However, such mobility patterns no longer reflect reality.

On the contrary, most of previous works in spatiotemporal database assumed that the movement pattern of a user closely approximated a line. For example, the MOST data model assumed that the movement pattern of real-life objects is very close to 1 dimension. That is, objects move in the plane but their movement is restricted on using a given set of routes on the finite terrain. This is called the 1.5 *dimensional problem*.

As we mentioned above, both MOD and PCS aim to study the movement patterns of real-life objects. Yet there is a difference in their assumption, approach, and environments. Therefore, more flexible and realistic model for the consideration of real-life mobility pattern is highly demanded. In MOD, real datasets of spatio-temporal patterns of real-life mobile users are very hard to obtain. Our approach is based on the *mobility-awareness* of location update protocol that split the whole movement of moving objects into the group of simple *movement state* (or simply called *state*). And, applying the corresponding update policy with a movement component dynamically minimizes the overall cost of the system.

3. State-based Mobility Model(SMM)

A mobility model, in the context of location management, is an understanding of daily movements of a user[9], and the description of this understanding. Motivated by this aim, various mobility models have been developed in mobile computing environments[8,9]. The mobility modeling in MOD is tricky by reason of the higher location granularity than that of PCS. Moreover, a matter of concern in MOD is not a logical/symbolic location, like *cell-id*, but the very physical/geographical location of moving object obtained by a location-sensing device such as GPS. As we mentioned before, it is essentially needed to consider a *complex* movement containing both a random and a linear movement patterns.

3.1 Basic Definitions

We model the *state-based mobility model* (SMM) that understands a complex mobility pattern as a set of simple movement components using a finite state Markov chain based on classification discussed in Section 2.

Definition 1. *A movement state* s_i *is a 3-tuple* $(v_{\min}, v_{\max}, \phi)$, *where* $v_{\min}$ *and* $v_{\max}$ *are the minimum and maximum speed of a moving object respectively.* ϕ *is a function of movement which is either probabilistic or non-probabilistic function.* S *is a finite set of movement states (called state space).*

Definition 2. *The state-based mobility model (SMM) describes a user mobility patterns using a finite state Markov Chain* $state_n$, *where* $state_n$ *denotes the movement state at step* n, $state_n \in S$. *And, the chain can be described completely by its* transition probability *as*

$$p_{ij} \equiv \Pr\{state_{n+1} = j | state_n = i\} \text{ for all } i, j \in S. \quad (1)$$

These probabilities can be grouped together into a transition matrix *as* $P \equiv (p_{ij})_{i,j \in S}$

An important question is why such a general mobility model is not as popular as the restrictive models so abundant in the literature[10]. The most important reason is that the generalized model has nothing to be assumed to start the analysis.

In this paper, we assume only that the whole mobility patterns are divided into three basic movement states such as *pause, linear,* and *random*. Each state has the self-transition probability p_{ii}. In the SMM model, we assume that a moving object has tendency to remain in the same state rather than switching states[8]. This is generally called *temporal locality*. The *self-transition probability*

vector (STPV) is obtained by taking all self-transition probabilities, which is equivalent to an $1\times|S|$ matrix.

Definition 3. *The self-transition probability vector (STPV)* $\tilde{\pi}$ *of a transition probability matrix is defined as* $\tilde{\pi}=(p_{ii})_{i\in S}$. *And the* ***temporal locality*** *(or locality)* τ *is defined as*

$$\tau=\left(\prod_{i\in S} p_{ii}\right)^{1/|S|}. \quad (2)$$

3.2 A Practical Instance of the SMM Model

As we mentioned before, the complex mobility patterns in the real world can be interpreted as a set of basic movement states (Figure 1). In this section, we present a practical instance of the proposed SMM model based on the three states described above, $S_0=\{P, L, R\}$.

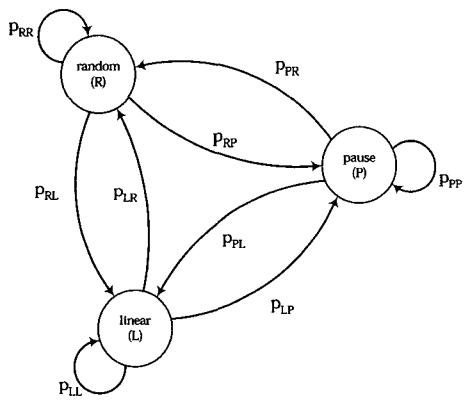

Figure 2 An Instance of SMM Model: $S_0=\{P, L, R\}$

Figure 2 shows a state transition diagram for this instance. Let us define two measurements that estimate how much each state has an influence on the whole movement pattern in this simplified model.

Definition 4. ***Linearity*** ℓ *is defined as*

$$\ell=\frac{\sum_{i\in S} P_{iL}}{\sum_{i,j\in S, j\neq L} P_{ij}} \quad (3)$$

Also, ***randomness*** γ *is defined as*

$$\gamma=\frac{\sum_{i\in S} P_{iR}}{\sum_{i,j\in S, j\neq R} p_{ij}} \quad (4)$$

For practical purpose, the above parameters is quite important to describe the various feature of mobility patterns.

4. State-based Location Update Protocol (SLUP)

4.1 Cost Model & Initial Experiment

Firstly, we assume the distributed location database in which the whole space is adequately decomposed into regions or cells. There is a location server (LS) that is a fixed host directly communicating with regional processors (RPs). As a lightweight server, each RP only monitors a small set of objects and transmits the information of its region to LS to maintain the consistency between LS and RP.

Suppose that there are a huge number of moving objects in d-dimensional space $\overrightarrow{R^d}$. For any time t, the position of the ith object is given by $o_i(t)$, which is a point in a d-dimensional space. Then, the movement history of the object o_i is described as a trajectory in $(d+1)$-dimensional space, which consists of $\langle o_i(0), o_i(1), \ldots, o_i(now)\rangle$. For location-dependent query processing, the LS should track the trajectory of network-registered moving objects. Thus an efficient protocol, which updates location information in the location server, is highly needed. The goal of a location update protocol is to provide more accurate location information with fewer update messages to LS. Clearly, this issue is a tradeoff between accuracy and efficiency.

Location update protocols are classified into four major classes in terms of when the update message is transmitted: time-based, movement-based, distance-based, and dead-reckoning[11]. Each update protocol has its own characteristics and different performance depending on its underlying mobility model. In other words, these algorithms need different amounts of update messages satisfying the same location precision or *uncertainty*.[1] We introduce a new criterion to compare the efficiency of update protocols using a simple formula by measuring the update cost and the imprecision cost for a certain amount of time. This criterion is called *UITR* (update-and-imprecision to time ratio) (see Eq. 6).

1) The term "uncertainty" is used as the threshold of location imprecision which is a difference between the actual location and database location. The concept of difference or distance is quite application-specific.

Naturally, it is preferable to keep the value of *UITR* small with providing the same location accuracy.

$$C_{UITR} = \frac{\sum_{k=1}^{wndsize} (w_U U_k + w_\varepsilon \varepsilon_k / \delta)}{wndsize} \quad (6)$$

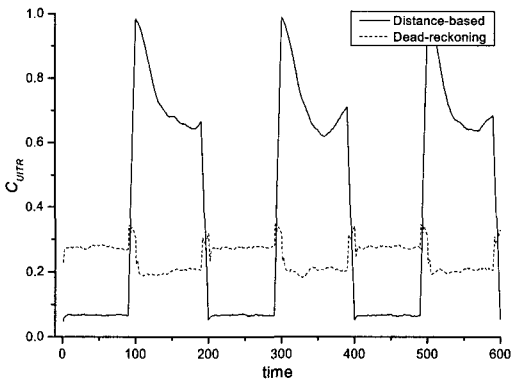

Figure 3 An example variations of C_{UITR}

To compute the value of *UITR* efficiently, we employ the update window (*UWin*) and the imprecision window (*IWin*) in the form of a circular queue. Each update flag (U_k) in *UWin* is true if update message is transmitted, or false if does not. Each item ε_k of *IWin* is the Euclidean distance between the actual location and the estimated location by an update policy. Both the *UWin* and the *IWin* contain *wndsize* cost items from the time t_i and the current time ($t_{i+wndsize}$). Therefore, the value of *UITR* is defined as the average cost of update and imprecision cost to the size of *UWin* and *IWin*. Figure 3 shows the comparison of distance-based approach and dead-reckoning approach in terms of C_{UITR}. In this comparison, the movement of whole objects is random or linear movements by turns with a fixed time interval 100.

We can identify the behavioral difference of two update policies in Figure 3. Exhibiting complementarity with respect to mobility patterns, different -- *mutually complement* -- update policies can be applied to the aforementioned states. Under the *pause* mode, we adopt the time-based approach to minimize the communication with base station. Reducing the average number of update messages can have a significant impact in power consumption of MT. In this way, it is important to find the reasonable threshold T, which is quite greater than in traditional approach. In the *linear* mode, as you know, the dead-reckoning approach has a great performance benefit especially in a constant speed. According to [3], the authors report that the update costs can be reduced by 83% compared to other protocols. A comprehensive study of dead-reckoning has been done in [3]. Finally, under the *random* mode, the movement patterns of an object have a special property of spatial locality. In this case, we employ the distance-based update protocol.

4.2 Protocol Details

During the life of a SLUP connection, the SLUP protocol running in each moving object makes transitions through various SLUP states. The behavior of a moving object is modeled as a state-transition diagram (Figure 4). Each moving object begins in the INITIAL state. The initialization process starts with the bootstrapping phase, then, carry out a self-test, performs the RP discovery by network layer functionality, and then enters the WARMUP state. While in the WARMUP state, each moving object will choose the beginning update policy according to their current mobility patterns. Exploiting temporal locality of mobility patterns, the update policy phase is decomposed into small fraction of update states such as UP_PAUSE, UP_LINEAR, and UP_RANDOM. Each update state consists of an update policy that is how the location information of an object is reflected in the location databases, the state-transition function determining the next states of the object, and information related to the state. The DISCONNECTED state is where a disconnection occurs between an object and the current RP. In this state, the last update policy with timestamp should be saved to local memory of an object. Upon reconnection, the state transition from DISCONNECTED to the saved update state is legal only if the time interval between the timestamp of saved update state and reconnection time is smaller than a given threshold T_{discon}. If not so, the destination of the state-transition will be the WARMUP state.

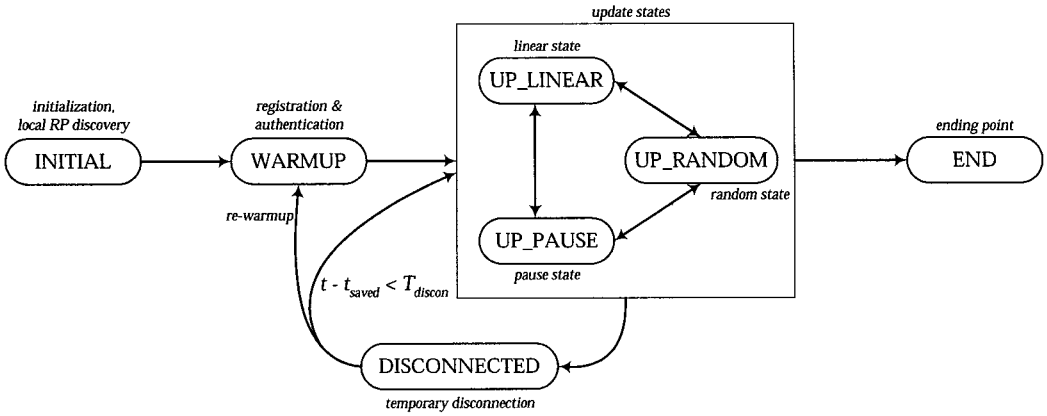

Figure 4 The state transition diagram for the SLUP protocol

As mentioned previously, each moving object performs not only the current update policy but also the others. Then the optimal update policy with the minimum *UITR* can be decided without any difficulty. The additional cost, a few memory and a small number of operation, is acceptable owing to its reflective effectiveness in the number of update messages and the development of hardware technology.

Definition 5. *The SLUP Protocol is based on the SMM model, and is represented by a finite set of update policy called update policy list* $UPL=\{\mu_1, \mu_2, \ldots, \mu_N\}$ *and the optimal update policy index opt.*

Definition 6. *An update policy* μ *is a 6-tuple* $(\hat{l}, f, C, UWin, IWin, \delta)$ *consisting of the estimated location* $\hat{l}$ *by* f*, a location estimation function* f*, the cost function* C*, the update window* $UWin$*, the imprecision window* $IWin$*, and a predefined location uncertainty* δ.

We provide a detailed algorithm in Figure 5. In this algorithm, the movement states are divided into two major classes: linear movements and random movements. The linear components can be abstracted as a linear function of time, which is specified by two parameters: the position and velocity vector for the object. Consequently, such an approach could minimize the location update cost peculiarly at a constant velocity. Whereas, we regard all other movements, including the random walk and ping-pong movements, as the random components. This is because it is difficult to

Algorithm *State-based Location Update Protocol*
Input: A set of update policies $UPL = \{\mu_P, \mu_L, \mu_R\}$
1. $t \leftarrow 0$
2. **repeat do**
3. **for** each state $\mu_i \in UPL$ **do**
4. $\hat{l}_i \leftarrow f_i(t)$
5. **if** $d(\hat{l}_i, l_{now}) > \delta_i$
6. **then** $UWin_i[t \bmod wndsize_i] \leftarrow true$
7. $\hat{l}_i \leftarrow l_{now}$ // pseudo-update
8. **else** $UWin_i[t \bmod wndsize_i] \leftarrow false$
9. $IWin_i[t \bmod wndsize_i] \leftarrow d(\hat{l}_i, l_{now})$
10. $C_i \leftarrow computeUITRCost(UWin_i, IWin_i)$
11. $opt \leftarrow argmin_{1 \le i \le N} C_i$
12. **if** μ_{opt} is pseudo-updated
13. **then** $SendUpdateMsg(opt, \hat{l}_{opt}, f_{opt}, \delta_{opt})$ to LS
14. $t \leftarrow t+1$
15. **until** satisfy termination condition

Figure 5 Algorithm for SLUP Protocol

discover certain knowledge or relevant information for further classification. This classification can be interpreted as a process of extracting the low-cost movement component from whole trajectory in term of their movement patterns. Even though simple, the approach that has never been tried in moving objects database is very efficient in complex movements.

5. Performance Evaluation

5.1 Simulation Model and Workload Generation

The workload for the scenario of moving objects can be either real, which extracted from the real-life applications, or synthetic by mathematical

model (e.g., probability theory). The artificially synthesized datasets are normally used to evaluate the impacts of specific parameters or in some case when the real datasets are not available. Since the real datasets in spatio-temporal database are very hard to achieve, the method of synthesizing data is widely used in various area [5-7]. In our experiment, we use two kinds of synthetic datasets: our proposed SMM model and City Simuator by IBM.

Using SMM Model. First of all, we suppose a centralized database for the evaluation of the proposed protocol with traditional update protocols. So, we only observe the number of sending the update messages with omitting the update step in RP layer. To simplify the simulation model, we assume that the delay time for sending messages is ignorable. We leave out the *pause* state because of the irrelevant with the evaluation. Then, we represent this assumption in the state-transition matrix, it is followed:

$$\mathbf{T}(\tau) = \begin{matrix} \\ L \\ R \end{matrix} \begin{matrix} \begin{matrix} L & \quad R \end{matrix} \\ \begin{pmatrix} \tau & 1-\tau \\ 1-\tau & \tau \end{pmatrix} \end{matrix}, \textit{where } \ell = 1 \text{ and } 0 \le \tau \le 1. \quad (7)$$

$$\mathbf{L}(\ell) = \begin{matrix} \\ L \\ R \end{matrix} \begin{matrix} \begin{matrix} L & \quad R \end{matrix} \\ \begin{pmatrix} \frac{\ell}{\ell+1} & \frac{1}{\ell+1} \\ \frac{\ell}{\ell+1} & \frac{1}{\ell+1} \end{pmatrix} \end{matrix}, \textit{where } 0 \le \ell \le \infty \text{ and } \tau = \sqrt{\frac{\ell}{(\ell+1)^2}}. \quad (8)$$

We classify the transition matrices for the probability into two types: $T(\tau)$ and $L(\ell)$. The type $T(\tau)$ has various characteristics with changing the locality τ from 0 to 1 within the linearity ℓ fixed with 1. The type $L(\ell)$ has the linearity ℓ and the same value for the L and R column. By definition, the locality of these matrices is $\sqrt{\ell/(\ell+1)^2}$. The maximum value of locality is 0.5, where $\ell = 1$ with omitting the pause state. For the simplicity, we represent the transition matrix of the type $T(\tau)$ with locality τ and the transition matrix of the type $L(\ell)$ with linearity ℓ as $T(\tau)$ and $L(\ell)$, respectively.

In the workload generation, we employ the parameters of the linearity, the locality and maximum speed of moving objects called v_{max} for our state transition matrix. The first mobility pattern is the pure random-walk. If the moving objects exist in the dimension $\overrightarrow{R^d}$ in this situation, it means that the probability of movement to all directions in dimension d is the same. These workloads may be generated by uniform, Zipf and Gaussian distribution like in[6]. If the linearity and locality are 0.0 and 1.0 respectively in the matrix, $L(0)$, we can generate these workloads. In our simulation, we use the movement vector by the real number extracted from uniform distribution within in the range of $[-v_{max}, +v_{max}]$ in each dimension.

On the other hand, we may consider a linear mobility pattern for all moving objects. In this situation, the trajectory of movements is almost straight line. We assume that the constant speed in this situation and the movement vector is generated by the same way with random state. These workloads are generated by the matrix L with the linearity ∞, $L(\infty)$. But the mobility patterns are more realistic if the two characteristics of movements, random-walk and linear mobility, are mixed appropriately because the mobility patterns for the real world may be both of all. Therefore, these mobility patterns are required for an approximate pattern to real world. We can create the workloads by changing the linearity and locality. Table 1 summarizes setting for the parameters in this experiment.

Table 1 Simulation Parameters for the SMM model

Parameter	Description	Values Used
#objects	the number of objects	1,000
v_{max}	maximum speed of moving objects	1
δ	location uncertainty	1.0, 2.0
#gen	the simulation time	1000
τ	the temporal locality	0.0 ~ 1.0 (spacing 0.1)
ℓ	the linearity of movement	0, 0.1, 0.25, 0.5, 0.75, 1, 5, 10, 20, ∞

Using City Simulator [12]. City Simulator is a scalable, three-dimensional model city that enables creation of dynamic spatial data simulating the motion of up to 1 million people moving along streets, buildings, and between building floors in three dimensions. Several parameters allow controlling the simulator. The *startThreshold* represents the fraction of people that should be on the ground level when the simulation should "start". And the *fillThreshold* represents the minimum fraction of people that should ever be left on the ground level. Finally, the *emptyThreshold* represents the maximum fraction of people that should ever be left on the ground level after a commute out of the buildings has been triggered. The parameters for City Simulator are presented in Table 2.

The simulated moving objects perform random walks on a building floor or they may move up or move down from one floor to the next floor depending on user-defined probabilities (*upProb* and *downProb*) if they are near to specific points (stairs)[12]. An object on a road moves with a linear combination of random walk and the drift velocity of the road; the influence of the drift velocity increases as a moving object gets closer to the center of a road.

5.2 Effect of Linearity ℓ

In this section, we discuss the effect of linearity ℓ. The experiment is performed on the transition matrix $L(0) \sim L(\infty)$. And the maximum speed v_{max} of all objects is 1. Thus, the pure random and the pure linear movements are represented as the transition matrices of $L(0)$ and $L(\infty)$ respectively.

Figure 6 shows the average number of update messages for an object with varying linearity ℓ and two different δ (1.0 and 2.0). In the distance-based approach, increasing the linearity gave rise to increasing the number of update messages. This is because the displacement of linear state is comparatively larger than random state in the same amount of time. On the other hand, the dead-reckoning approach performs very well in the case of increasing linearity. Above all, the performance has increased considerably when the parameter linearity bigger than 1.0. The

Table 2 Simulation Parameters for City Simulator Datasets

City Simulator dataset	0	1	2	3	4	5	6	7	8	9	10
fillThreshold	0.00	0.01	0.03	0.05	0.07	0.09	0.18	0.27	0.36	0.45	0.54
startThreshold	0.00	0.03	0.06	0.09	0.12	0.15	0.24	0.33	0.42	0.51	0.60
emptyThreshold	0.25	0.30	0.35	0.40	0.45	0.50	0.60	0.70	0.80	0.90	0.99
NumPeople	1,000	The number of people simulated.									
MaxRelax	2,000	Maximum number of relaxation steps before start of simulation. The simulator iterates until MaxRelax cycles or until the startThreshold is reached.									
FinalSteps	1,000	The number of simulation cycles actually stored in the datafile.									

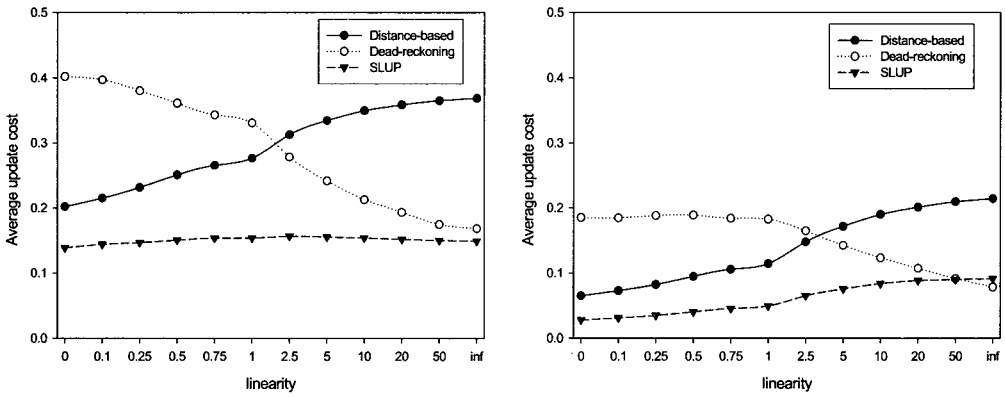

Figure 6 Effect of linearity ℓ: δ=1.0 and δ=2.0

proposed approach has outperformed traditional approaches in the every case of varying linearity except $L(\infty)$. The matrix $L(\infty)$ implies that all objects moves along a straight line without changing any direction vector.

5.3 Effect of Locality τ

In this section, similar to Section 5.2, we discuss the impact of varying locality. For example, if locality is 0.0, the object will change to other state unconditionally. On the other hand, if locality is 1.0, the object will maintain its current state forever. This locality has a great influence on the performance of location update protocol. The experiment is performed on the transition matrix $T(0) \sim T(1)$. And the maximum speed v_{max} of all objects is 1. The matrix $T(\tau)$ can be defined as a state-transition matrix with fixed linearity 1.0 and varying locality τ. In respect of the matrix, the quantity of random movements is identical with that of linear movements, for the linearity is fixed to 1.0. However, a transition matrix with larger locality is likely to have more linear movements than the opposite one. The dead-reckoning approach, therefore, will be advantageous for a larger locality under the same linearity.

Figure 7 shows the average number of update messages for an object with varying locality τ and δ parameters. In the distance-based approach, increasing the locality gave rise to increasing the number of update messages. Since the linearity is fixed to 1.0, such performance degradation in the previous section can be avoided. Like the previous results, moreover, the performance of proposed approach is likely to have the same curve as that

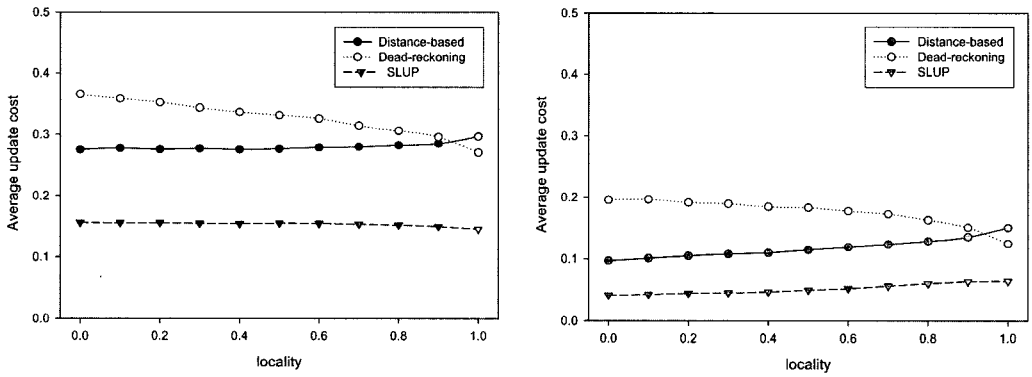

Figure 7 Effect of locality τ: δ=1.0 and δ=2.0

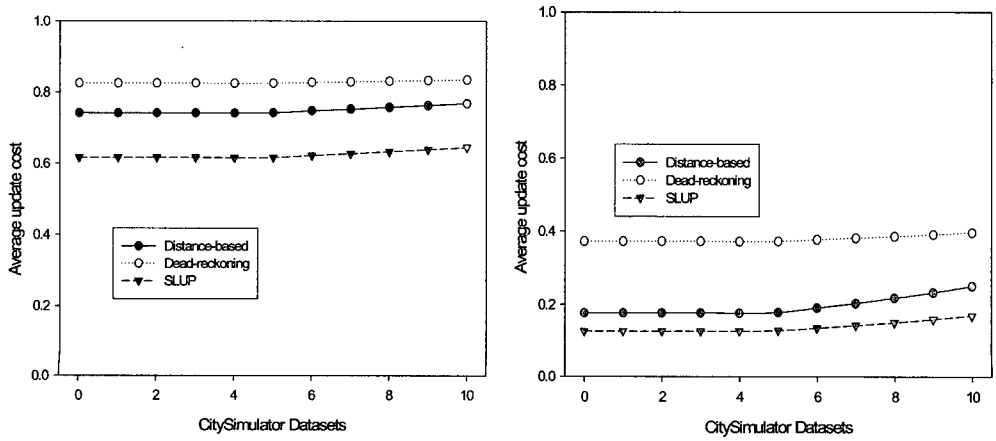

Figure 8 Datasets from City Simulator : δ=1.0 and δ=2.0

of dead-reckoning approach. The proposed approach has outperformed than the dead-reckoning approach in the every case of varying linearity.

5.4 Using the Datasets from City Simulator

In order to reexamine the superiority of proposed scheme, we consider an external spatiotemporal dataset generator called City Simulator developed by IBM[12]. As described in Tabe 2, we haved generated 11 datasets by changing three threshold probabilities: *fillThreshold*, *startThreshold*, and *emptyThreshold*. At this experiment, we simplify the simulation by considering only (x,y) position among the (x,y,z) location generated from City Simulator.

Figure 8 shows the average number of update messages for an object with two different δ parameters (1.0 and 2.0). In this experiment, our proposed approach has outperformed traditional approaches under all conditions. Particularly, the location update cost is slightly increased from the 5th dataset in which *fillThreshold*=0.09, *startThreshold*=0.15, and *emptyThreshold*=0.50. This is because the number of randomly moving object on ground level increases as the value of startThreshold probability increases. In summary, the maximum performance improvements for $\delta=1.0$ and $\delta=2.0$ made by the proposed scheme are 25% and 66% respectively.

6. Conclusions and Future Work

In this paper, we argue that a generalized mobility model is quite effective to model the location and movement of moving objects. In order to provide efficient location update strategy, we have proposed a new mobility model called SMM to describe movement patterns of real-life objects in probabilistic manners. This approach outperforms in the mixed situation with linear movements and random movements.

Future researches include the following. Firstly, we want to investigate more useful property of proposed SMM model based on Markov chain theory. Secondly, we develop a highly scalable indexing structure for moving objects environment. Finally, in the future, we plan to implement a portable software interface for the workload generation based on our SMM model.

References

[1] Computer Science and Telecommunications Board (CSTB), *National Research Council. IT Roadmap to a Geospatial Future*, The National Academies Press, 2003.

[2] Ralf Hartmut Guting, Michael H. Bohlen, Martin Erwig, Christian S. Jensen, Nikos A. Lorentzos, Markus Schneider, and Michalis Vazirgiannis. "A Foundation for Representing and Querying Moving Objects," *ACM Transactions on Database Systems*, 25(1), 2000.

[3] O. Wolfson, A. P. Sistla, S. Chamberlain, and Y. Yesha, "Updating and querying databases that track mobile units," *Distributed and Parallel Databases*, 7(3):257--387, 1999.

[4] Ouri Wolfson, "Moving Objects Information Management: The Database Challenge," *Proc. of NGITS*, 2002.

[5] T. Brinkhoff, "Generating network-based moving objects," *Proc. of SSDBM*, 2000.

[6] D. Pfoser and Y. Theodoridis, "Generating semantics-based trajectories of moving objects," *Proc. of Workshop on Emerging Technologies for Geo-Based Applications*, 2000.

[7] J.-M. Saglio and J. Moreira, "Oporto: A realistic scenario generator for moving objects," *GeoInformatica*, 5(1):71--93, 2001.

[8] Y.-C. Tseng, L.-W. Chen, M.-H. Yang, and J.-J. Wu, "A stop-or-move mobility model for PCS networks and its location-tracking strategies," *Computer Communications*, 26:1288-1301, 2003.

[9] T. Kunz, A. A. Siddiqi, and J. Scourias, "The peril of evaluating location management proposals through simulations," *Wireless Networks*, 7(6): 635-643, 2001.

[10] A. Bhattacharya and S. K. Das, "LeZi-update: An information-theoretic approach to track mobile users in PCS networks," *Proc. of MOBICOM*, 1999.

[11] A. Bar-Noy, I. Kessler, and M. Sidi, "Mobile users: To update or not to update?," *Wireless Networks*, 1(2):175--186, 1995.

[12] IBM alphaWorks: City Simulator, http://www.alphaworks.ibm.com/tech/citysimulator

[13] E. Pitoura and G. Samaras, "Locating Objects in Mobile Computing," *IEEE Transaction on Knowledge and Data Engineering*, 13(4), 2001.

[14] D. Barbara, "Mobile computing and databases - A survey," *IEEE Transaction on Knowledge and Data Engineering*, 11(1):108--117, 1999.

송 문 배

1996년 군산대학교 이학사. 1998년 숭실대학교 이학 석사. 2003년 고려대학교 이학 박사 수료. 2001년~현재 고려대학교 컴퓨터과학기술 연구소 연구원. 관심분야는 모바일 컴퓨팅, 이동객체 데이타베이스, 시공간 데이타 마이닝

강 상 원

1998년 고려대학교 전산학 학사. 2003년 고려대학교 컴퓨터학 석사. 2003년~현재 고려대학교 컴퓨터학 박사과정. 관심분야는 모바일 데이타관리, LBS, 이동 객체, 데이타 마이닝, 퍼베이시브 컴퓨팅

박 광 진

2000년 고려대학교 이학 학사. 2002년 고려대학교 이학 석사. 2004년 고려대학교 이학 박사 수료. 2004년~현재 고려대학교 컴퓨터과학기술 연구소 연구원. 관심분야는 모바일 컴퓨팅, 캐시 관리, 위치 기반 서비스

황 종 선

1978년 Univ. of Georgia, Statistics and Computer Science 박사. 1978년 South Carolina Lander 주립대학교 조교수. 1981년 한국표준연구소 전자계산실 실장. 1995년 한국정보과학회 회장. 1982년 ~ 현재 고려대학교 컴퓨터학과 교수 1996년 ~ 현재 고려대학교 컴퓨터과학기술대학원 원장. 관심 분야는 알고리즘, 분산시스템, 데이타베이스, 이동컴퓨팅 등