Mobile Information Systems 1 (2005) 109–127
IOS Press

On the design of energy-efficient location tracking mechanism in location-aware computing[1]

MoonBae Song*, Sang-Won Kang and KwangJin Park
Distributed Systems Laboratory, Department of Computer Science & Engineering, Korea University 1, 5-Ga, Anam-Dong, Sungbuk-Gu, Seoul 136-701, Korea

Abstract. The battery, in contrast to other hardware, is not governed by Moore's Law. In location-aware computing, power is a very limited resource. As a consequence, recently, a number of promising techniques in various layers have been proposed to reduce the energy consumption. The paper considers the problem of minimizing the energy used to track the location of mobile user over a wireless link in mobile computing. Energy-efficient location update protocol can be done by reducing the number of location update messages as possible and switching off as long as possible. This can be achieved by the concept of *mobility-awareness* we propose. For this purpose, this paper proposes a novel 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. An extensive experiment on various synthetic datasets shows that the proposed method improves the energy efficiency by $2 \sim 3$ times with the additional 10% of imprecision cost.

Keywords: Location update protocol, energy efficiency, moving objects database, mobile computing, location-aware computing

1. Introduction

In mobile 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] [6,16, 17]. Thus, one of the most important issues in mobile computing is how to model and track the location and the movement of moving objects efficiently [11]. Therefore, a software infrastructure for providing location information, called *moving objects database* (MOD), is significantly needed.

The "doubling time" of hardware such as microelectronics, memory, and secondary storages is governed by so-called *Moore's Law*. Although such an improvement on computer hardware, the lifetime of a battery is expected to increase only 20% over the next 10 years [7]. As a result, attention to power

*Corresponding author: MoonBae Song. Tel.: +82 2 924 0547; Fax: +82 2 953 0771; E-mail: mbsong@disys.korea.ac.kr.
[1]A preliminary version of this paper has appeared in The Sixth International Conference on Information Integration and Web-based Applications & Services (iiWAS2004), Jakarta, Indonesia, September 27–29, 2004.
[2]The term "moving object" is used as a more general term/concept. So, it may be mobile clients, car, air plane, soldiers in battle field, wildfire, hurricanes, and so forth. In this paper, for the theoretical reason, moving objects (MOs) is more prefer than mobile clients (MCs).

consumption must span many levels of hardware and software to be fully effective. Especially, the location update protocol is ever-running process for MCs. Therefore, it is very important to achieve energy efficiency. Basically, the energy efficiency in location update protocol can be done by

– *Reducing location update message as possible*: Transmitting a message consumes more power than receiving. Moreover, this power grows as the fourth power of the distance between the client and the server [7]. In the viewpoint of location update policy, the uplink message is issued when the difference between actual location and database location exceeds a predefined threshold δ.[3] Reducing the imprecision is, therefore, reducing the number of update messages as possible. Such a "laziness" to the location update is very helpful to the energy efficiency.
– *Switching to doze mode as possible*: By exploiting stationary state, location tracking for a stationary host can be done in doze mode. The *doze mode* has extremely low power consumption. The network interface card (NIC) can neither transmit nor receive until it is woken up. Then, how to decide when an MC switch-off itself? For this purpose, an MC should be aware of its movement.

Eventually, an MC should be aware of its own mobility. This is what we called *mobility-awareness*. In order to be aware of the mobility, we have to develop a model which describes a per-user movement. This is called *mobility modeling*. In this paper, we look at the mobility model for MOD and an appropriate location update protocol. The purpose of our model is to model the overall movement patterns in a probabilistic manner. In a previous work [14], a preliminary model for location tracking and modeling was presented. This paper extends our previous work to analyze and reduce the energy consumption of MCs.

Recently, there is a lot of work on the representation and management of moving objects [6,16,17]. Wolfson et al. present the well-known data model called Moving Object Spatio-Temporal (MOST) for representing moving objects [17]. In this model, the location of moving objects is simply given as a linear function of time, which is specified by two parameters: the position and velocity vector for an object. Thus without frequent update message, the location server can compute the location of a moving object at given time t by linear interpolation: $y(t) = y_0 + \bar{v}(t - t_0)$ at time $t > t_0$. The update message is only issued when the parameter of linear function, e.g. $\bar{v}$, changed. In general, we say that this update approach is *dead-reckoning*. The dead-reckoning approach can provide a great performance benefit in linear mobility patterns. But the performance is decreased when the randomness of mobility pattern increases. Another major drawback is the inaccuracy of the predicted location by linear interpolation.

To the best of our knowledge, very few contributions to energy-efficiency in location tracking mechanism have been reported in the literature. In existing approaches, MCs always check where they are and whether or not to transmit a location update message in each time unit. Therefore, MCs cannot doze-off its processor, and switch off NIC.

The remainder of this paper is organized as follows. In the next section, we discuss the location update policies in location-aware computing. In Section 2.1, we introduce the characteristics of mobility patterns of real-life objects. Then, the proposed mobility model, called *state-based mobility model* (SMM), will be described. 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. The 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.

[3]The database location is the location information stored in the database. The difference and its threshold are called "imprecision" and "uncertainty" respectively.

2. Modeling user movement

A mobility model, in the context of location management, is an understanding of daily movements of a user and formal description of this understanding. 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. Thus, a mobility model in MOD is essentially needed to consider a *complex* movement containing both a random and a linear movement patterns.

2.1. 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 [3,4,15,17]. 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'.

Basically, our scheme was motivated by the observation that the behavior of real-life moving objects can be interpreted as a set of movement components (repeatedly) such as linear, pause, and random. For example, in the *linear* movements, the trajectory of an object is almost a line in d-dimensional space. Otherwise, if we can't find the implicit knowledge of a specific pattern, let us identify the portion of trajectory as a *random walk* or a *Brownian movement*. And we consider the temporal pattern of movements as Markovian process.

2.2. Basic definitions

In this section, we propose the basic definitions of *state-based mobility model* (SMM) considering a complex mobility pattern as a set of simple movement components using a finite state Markov chain based on the classification discussed in Section 2.1.

Definition 1. (Movement State) 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.

Definition 2. (SMM Model) 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 $p_{ij} \equiv Pr\{state_{n+1} = j | state_n = i\}$ for all $i, j \in S$. These probabilities can be grouped together into a *transition matrix* as $\mathbf{P} \equiv (p_{ij})_{i,j \in S}$, where $\sum_{j \in S} p_{ij} = 1 \quad \text{for all} \quad i \in S$.

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 also assume that a moving object has tendency to remain in the same state rather than switching states [18]. This is generally called *temporal locality*.

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

$$\tau = \left(\prod_{i \in S} \left(\begin{cases} 1, & \text{if } p_{ii} = 0; \\ p_{ii}, & \text{otherwise.} \end{cases} \right) \right)^{1/|S|}, \tag{1}$$

where $|.|$ denotes the cardinality of a set.

2.3. 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. The most definite movement states are *pause*, *linear movement*, and *random movement*. In this section, we present a practical instance of the proposed SMM model based on the three states described above, the state space $S_0 = \{\mathsf{P} \equiv pause, \mathsf{L} \equiv linear, \mathsf{R} \equiv random\}$. Let us define a measurement that estimates how much each state has an influence on the whole movement patterns in this simplified model.

Definition 4. (Dominancy) The dominancy η_s of state $s \in S$ is defined as

$$\eta_s = \frac{\sum_{i \in S} p_{is}}{\sum_{i,j \in S, j \neq s} p_{ij}}.$$

As a special case of *dominancy*, the η_P, η_L, and η_R are called *stationarity* (ρ), *linearity* (ℓ), and *randomness* (γ) respectively. For a practical purpose, the above parameters are quite important to describe the various features of mobility patterns.

2.4. Determining the transition matrix

One of the most important things in the SMM model is to determine the transition matrix $\mathbf{P}$. The matrix can be determined by the user profile, the spatiotemporal data mining process, or in an ad hoc manner. The simplest way to determine the transition matrix $\mathbf{P}$ is to set $p_{ij} = \frac{1}{|S|}$ for all $i, j \in S$. A more reasonable solution is to use statistical techniques to infer the values of the transition probabilities empirically from past data [2,10]. For example, suppose the optimal state for each time unit is a Markov chain having state space S_0. Furthermore, assuming that the optimal state[4] for 36 time units has been "PPPLLLLRRLPLLLRRRPPPRLLRLLLLRPPRRRRP". Counting the number of transitions N_{ij}[5]

[4]The meaning of *optimal state* is a quite application-specific. For instance, the meaning of this term is a minimal-cost of location management/update, where the precision function of location information associated with each state is need to satisfy a threshold called location uncertainty δ.

[5]We only assume first-order Markov chain model. So, N_{ij} is defined as

$$N_{ij} = \sum_{k=1}^{m-1} \left(\begin{cases} 1, & \text{if } o_k = i \text{ and } o_{k+1} = j \\ 0, & \text{otherwise.} \end{cases} \right),$$

where a sequence of optimal states $\mathcal{O} = o_1 o_2 \ldots o_m$.

from state i to state j gives

$$[N_{ij}]_{i,j\in S} = \begin{array}{c} \\ \mathsf{P} \\ \mathsf{L} \\ \mathsf{R} \end{array} \begin{array}{c} \begin{array}{ccc} \mathsf{P} & \mathsf{L} & \mathsf{R} \end{array} \\ \begin{pmatrix} 5 & 2 & 2 \\ 1 & 9 & 4 \\ 3 & 3 & 6 \end{pmatrix} \end{array}. \tag{2}$$

The transition probability p_{ij} can now be estimated as $\hat{p}_{ij} = \frac{N_{ij}}{\sum_{k\in S} N_{ik}}$ giving the maximum-likelihood estimator of $\mathbf{P}$ as

$$\hat{\mathbf{P}} = \begin{array}{c} \\ \mathsf{P} \\ \mathsf{L} \\ \mathsf{R} \end{array} \begin{array}{c} \begin{array}{ccc} \mathsf{P} & \mathsf{L} & \mathsf{R} \end{array} \\ \begin{pmatrix} 0.5556 & 0.2222 & 0.2222 \\ 0.0714 & 0.6429 & 0.2857 \\ 0.2500 & 0.2500 & 0.5000 \end{pmatrix} \end{array}. \tag{3}$$

From this estimated matrix, we can calculate the movement parameters such as $\ell = 0.59160$, $\gamma = 0.50595$, $\rho = 0.41309$, and $\tau = 0.56320$ by Definition 3 and 4. Let $\boldsymbol{\pi} = (\pi_P, \pi_L, \pi_R)$ be the *steady-state* probability vector or *equilibrium* [10]. Solving for $\boldsymbol{\pi} = \boldsymbol{\pi}\times\mathbf{P}$, we obtain $\pi_P = 0.257$, $\pi_L = 0.4$, and $\pi_R = 0.343$ as estimates of residence probabilities of each movement state [10]. By means of this statistical technique, the reasonable estimation of $\mathbf{P}$ can be estimated from user's past movement. Obviously, it is possible to dynamically instantiate a SMM model corresponding to the realistic movement.

3. Preliminaries

3.1. *Energy-efficient approaches in application layer*

In this section, we begin by quoting the insight of Imielinski et al. for the design of energy-efficient approach for mobile computing environments [7, p. 355]:

> "The ability to selectively switch off the receiver and avoid transmitting as much as possible will be very important to conserve battery power ... Cordless phones listen to the base stations only for a small fraction of time and they keep their receivers switched off most of the time. A slight delay in receiving a call is not noticeable by humans. This idea of switching off the receiver most of the time increases the battery life of cordless phones from hours to a week. Similar ideas can be used for palmtops and data transmission in general."

Energy-efficient technique is a class of adaptation technique for mobile computing environment in which mobile clients have scare resources such as limited battery, bandwidth, computation power. Energy efficiency at the application layer is becoming increasingly important area of research. Most of existing works have focused on *load partitioning* in which applications may be selectively partitioned between the mobile client and base station [8].

Also, like other layers, another technique for conserving energy is to turn off the transceiver most of time and doze off the processor selectively. Imielinski et al. [7] propose so-called "air indexing" scheme (to reduce the energy consumption of mobile clients) in which a data server periodically broadcasts a pre-computed directory information that contains data transmission starting times for each MC. Thus, MCs can stay in the *power saving* mode most of the time and tune into the broadcast channel only when the requested data arrives. This is called *selective tuning* mechanism [7].

In addition, caching frequently accessed data at the mobile clients can also save the energy and bandwidth used to retrieve the repeatedly requested data at client side. When the data items are available in the cache, mobile clients can reduces not only the uplink and downlink bandwidth consumption but also the average data access delay. Cai and Tan propose three cache invalidation schemes [5]. In these approaches, like in [7], the server periodically broadcasts object invalidation reports (IR). And clients can selectively tune to the portion of the invalidation reports that are of interest to them. So, this allows the clients to increase the cache hit ratio, and consequently the energy consumption of MC will be reduced.

Jones et al. provides a good survey of energy efficient network protocol for each layer of the protocol stack [8].

3.2. Location update policies in location-aware computing

Location update policy is a compromise between the location update cost and the precision of location information stored in the location database. Because of the continuous movement of a moving object, the location information is inherently imprecise regardless of the policy used to update the database location of the object [17].

Suppose that there are a huge number of moving objects in d-dimensional space $\mathbb{R}^d = [-\infty, +\infty]^d$. 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 location server (LS) should track the trajectory of network-registered moving objects. Thus, an efficient protocol, which updates their 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 has a tradeoff between accuracy and efficiency. The more update messages we transmit, the more precise location information we get. In spite of this tradeoff relationship, each policy will offer a different opinion for whether the current location information has to transmit to the server or not.

The abstracted form of location update policy is as follows:

```
procedure LocationUpdatePolicy() {
    forever do {
1:     Every $t_{sample}$ time, get the actual location
2:     Estimate the current location using an estimation
          function based on a location update message
          previously reported
3:     Compare the actual location with estimated location
          based on a constraint
4:     if a trigger condition is true
5:        then send update message
    }
}
```

And the classes of location update policies and their relationships are shown in Fig. 1 on the basis of estimation function (step 2) and trigger condition (step 4).

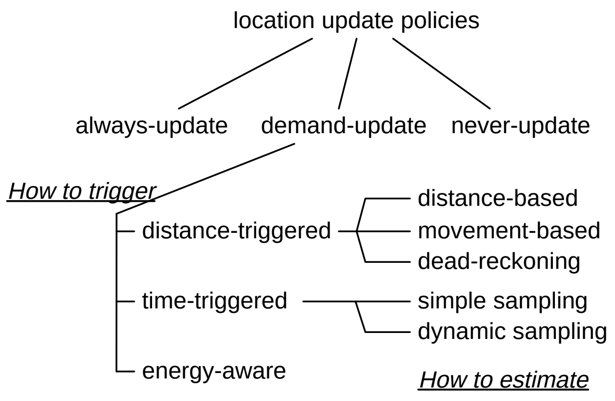


Fig. 1. The taxonomy of location update policies.

The simplest approach is the periodic sampling of a location source (*always-update policy*). In this way, redundant update messages and server overload is inevitable. In other extreme, an approach is called *never-update policy*, if the LS decides when to request the location information from the source. In *demand-update policy*, transmitting the location update message is performed on demand along with three trigger-points: distance, time, and energy. Such update policies are classified into four major classes in terms of when the update message is transmitted: time-based, distance-based, movement-based, and dead-reckoning [3]. In time-based policy, each MC transmits an update message every $\hat{T}$ time unit. Distance-based policy sends an update message whenever the (geographic or symbolic) distance between the current location and the last updated location exceeds a given threshold D. Movement-based policy interpreted as an accumulated version of the distance-based policy. The dead-reckoning is a predictive version of distance-based policy exploiting the *linear interpolation*: $y(t) = y_0 + \bar{v}(t - t_0)$ at time $t > t_0$. And the distance-based policy can be interpreted as a special case of dead-reckoning, i.e. $\bar{v} \rightarrow 0$. Each update protocol has its own characteristics and different performance behaviors depending on the underlying mobility model. In other words, these algorithms need different amounts of update messages satisfying the same location precision or *uncertainty*.

Recently, there is a lot of work on the representation and management of moving objects [6,16,17]. Wolfson *et al.* present the well-known data model called Moving Object Spatio-Temporal (MOST) for representing moving objects [17]. In this model, the location of moving objects is simply given as a linear function of time, which is specified by two parameters: the position and velocity vector for an object. Thus without frequent update message, the location server can compute the location of a moving object at given time t by linear interpolation: $y(t) = y_0 + \bar{v}(t - t_0)$ at time $t > t_0$. The update message is only issued when the parameter of linear function, e.g. $\bar{v}$, changed. In general, we say that this update approach is *dead-reckoning*. The dead-reckoning approach can provide a great performance benefit in linear mobility patterns. But the performance is decreased when the randomness of mobility pattern increases. Another major drawback is the inaccuracy of the predicted location by linear interpolation.

To the best of our knowledge, very few contributions to energy-efficiency in location tracking mechanism have been reported in the literature. In existing approaches, MCs always check where they are and whether or not to transmit a location update message in each time unit. Therefore, MCs cannot doze-off its processor, and switch off network interface card (NIC).

4. The proposed protocol

4.1. Cost model and initial experiment

Firstly, 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 $\mathcal{UITR}$ (*update-and-imprecision to time ratio*) (Eq. 4).

$$C_{\mathcal{UITR}}(\Delta, w_u, w_\varepsilon) = \frac{\sum_{k=1}^{\Delta}(w_u u_k + w_\varepsilon \frac{\varepsilon_k}{\delta})}{\Delta} \tag{4}$$

To compute the value of $\mathcal{UITR}$ 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 1 if an update message is transmitted, or 0 if it does not. Each item ε_k of $IWin$ is the Euclidean distance between the actual location and the estimated location by an update policy. For the computation of $C_{\mathcal{UITR}}$, ε_k is normalized by the predefined uncertainty δ. Both the $UWin$ and the $IWin$ contain Δ cost items from the time t_i and the current time ($t_{i+\Delta-1}$). The relative importance between update cost and imprecision cost in the cost model is controlled by two weights w_u and w_ε. The $C_{\mathcal{UITR}}$ model can be interpreted as an integration of location update cost and imprecision cost, and an indirect approach to measure energy consumption by setting $w_u \gg w_\varepsilon$.

Based on this cost model, we can identify the behavioral difference of existing location update policies. Figure 2 shows the comparison of distance-based approach, dead-reckoning, and time-based approach in terms of $C_{\mathcal{UITR}}$ model. In this comparison, the movement of whole objects is random-walk or linear mobility patterns by turns with a fixed time interval 200. For random-walk, the distance-based approach outperforms the dead-reckoning in which the frequent update of motion vector may cause the increased location imprecision and the resulting increased update cost as well. For linear movement, in contrast, the dead-reckoning approach performs very well. But, the performance of the distance-based approach extremely degraded, since average movement distance per unit time highly increased.

This initial experiment shows that the *mobility-awareness* is the key to reducing the number of location update messages. Exhibiting complementarity with respect to mobility patterns, different – *mutually complement* – update policies can be applied to the aforementioned states: pause (P), linear (L), and random (R).

4.2. Protocol details

After power-up, each moving object makes transitions through various states depending on its mobility patterns and terminal states, and updates its location information in location databases using the current update policy (Fig. 3). The object should examine the $\mathcal{UITR}$ value of all update policies in order to determine an optimal policy over the whole set of update policies.

The behavior of a moving object is modeled as a state-transition diagram. Each moving object begins in the INITIAL state. The initialization process starts with the bootstrapping phase, then, carries 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 state* such as UP PAUSE, UP LINEAR, and

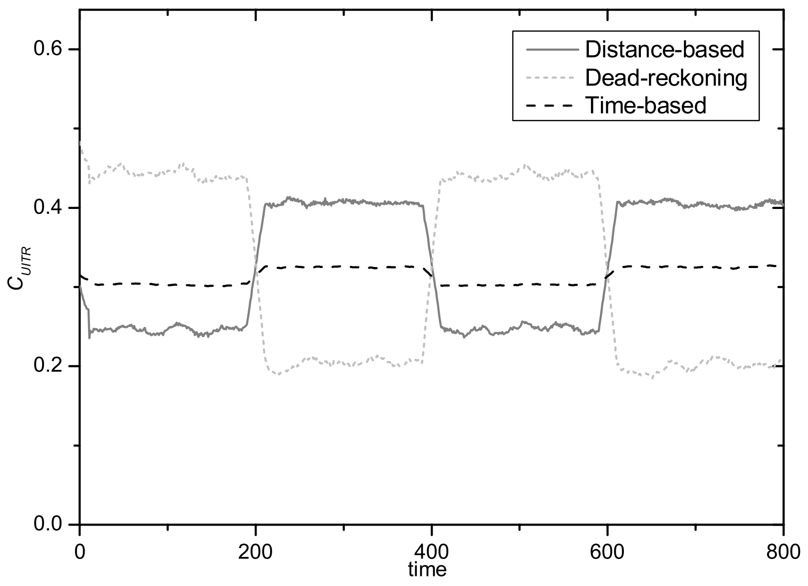


Fig. 2. Initial Experiment: An example variations of $C_{\mathcal{UITR}}(5, 1, 0.1)$.

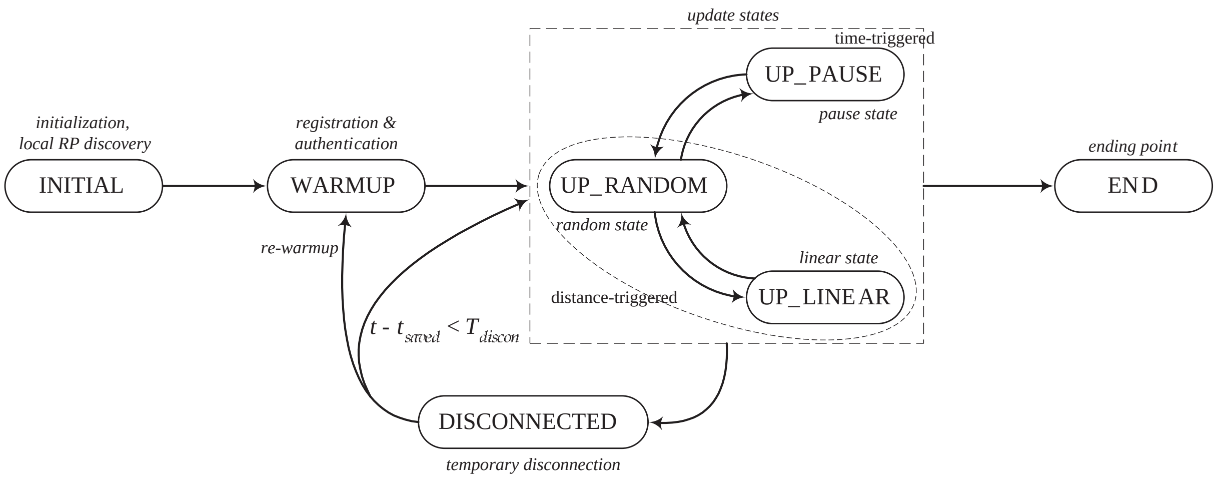


Fig. 3. The state transition diagram for the SLUP protocol.

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.

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 $\mathcal{UITR}$ can be decided without any difficulty. The additional cost, a few memory and a small number of operations, is acceptable owing to its reflective effectiveness in the number of update messages and the development of hardware technology.

Definition 5. (State-based Location Update Protocol) The SLUP Protocol is based on the SMM model and is represented by a finite set of update policy called *update policy list* $\mathcal{UPL} = \{\mu_1, \mu_2, \ldots, \mu_N\}$ and

Table 1
Behaviors of each device in terms of various energy states of proposed protocol

Energy state\device	Processor	NIC	GPS
DOZE	Doze-off	Switch-off	Power saving
IDLE	Normal	Idle	Continuous
TRANSMIT	Normal	Transmit	Continuous

the optimal update policy index *opt*.

Definition 6. (Update policy) 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 Fig. 4. From the starting state R, location update is performed with state transition through P, L, and R. If the optimal state is P, then the SLUP protocol will be in time-triggered manner in period $\hat{T}$ and be in the doze mode in the remainder of time. Otherwise, it will performs a distance-triggered manner with distance threshold δ. If the opt state is stale, then call $ChooseOptimal()$ to choose a new (predicted) optimal state and transmit location update message.[6] Otherwise, check the current opt is R and still remains for a certain amount of time T_p, then state station to the state P is performed.

To the best of our knowledge, few studies on the location update policy for stationary state found in the literature. For this reason, every MC should operate even in stationary state. Under the pause mode, the moving object is perhaps located at his/her home, office, or meeting room for a long time. It is quite reasonable to consider a location update policy for stationary state as well as moving state. This insight has been tried in the location management of PCS, called *Stop-or-Move* mobility model [18]. We can assume that an object is in a stationary state if the moving object remains still its location for a certain amount of time T_p. Then, in this "doze mode", it is great benefit in terms of the power consumption of mobile clients by turning off the transceiver and dozing off the processor. The proposed approach performs location updates and makes transitions through different energy states depending on user's movement. These energy states are DOZE, IDLE, and TRANSMIT. In Table 1, behavioral differences of each device in terms of various energy states is presented.

5. Performance evaluation

5.1. Simulation model and workload generation

We evaluate the performance of three traditional approach, and the proposed approach with variation. The traditional approaches include distance-based, time-based with threshold($\hat{T}_{time}$) 3, and dead-reckoning. We consider a range of dominancy and temporal locality. The performance metrics include both energy cost and average imprecision.

Since the real datasets in spatio-temporal database are very hard to achieve, the method of synthesizing data has widely been used in various area [4,15]. By exploiting the SMM model in Section 2, we can

[6] A state is called *stale* (or inconsistent) if the location information of a state has more location imprecision than a predefined uncertainty δ. In this paper, we assume that $ChooseOptimal()$ is return state R if current optimal state is L and vice versa.

```
in UP_PAUSE state {
    Stay in doze mode and wake up every $\hat{T}$ time unit;
    if $(dist(\hat{l}_{\mathsf{P}}, l_{now}) > \delta_{\mathsf{P}})$
        Switch into R state and send update message
            $\langle \mathsf{R}, l_{now}, \mathbf{null}, \delta_{\mathsf{R}}, t \rangle$;
}
in UP_LINEAR state {
    $PseudoUpdate(\mu_{\mathsf{R}}, t)$;
    if $(dist(\hat{l}_{\mathsf{L}}, l_{now}) > \delta_{\mathsf{L}})$ {
        // opt is stale. Do real update.
        $PseudoUpdate(\mu_{\mathsf{L}}, t)$;
        Choose a new state by calling ChooseOptimal();
        Switch into the new state and send update message
            $\langle new, \hat{l}_{new}, f_{new}, \delta_{new}, t \rangle$;
    }
}
in UP_RANDOM state {
    $PseudoUpdate(\mu_{\mathsf{L}}, t)$;
    if $(dist(\hat{l}_{\mathsf{R}}, l_{now}) > \delta_{\mathsf{R}})$ {   // opt is stale. Do real update.
        $PseudoUpdate(\mu_{\mathsf{R}}, t)$;
        Choose a new state by calling ChooseOptimal();
        Switch into the new state and send update message
            $\langle new, \hat{l}_{new}, f_{new}, \delta_{new}, t \rangle$;
    }
    else if there is no location update during last $T_p$ time units {
        // switch into UP_PAUSE state.
        $PseudoUpdate(\mu_{\mathsf{P}}, t)$;
        Switch into P state and send update message
            $\langle \mathsf{P}, \hat{l}_{\mathsf{P}}, \mathbf{null}, \delta_{\mathsf{P}}, t \rangle$;
    }
}

$PseudoUpdate(\mu_i, t)$ {
    // a self-initiated update procedure within an MC's local memory
    Update $\hat{l}_i$ by the location estimation function $f_i(t)$;
    if $(dist(\hat{l}_i, l_{now}) > \delta_i)$
        $UWin_i[t \ \mathbf{mod}\ \Delta] \leftarrow 1$; $\hat{l}_i \leftarrow l_{now}$;
    else $UWin_i[t \ \mathbf{mod}\ \Delta] \leftarrow 0$;
    $IWin_i[t \ \mathbf{mod}\ \Delta] \leftarrow dist(\hat{l}_i, l_{now})$;
    $C_i \leftarrow computeUITRCost(UWin_i, IWin_i)$;
}
```

Fig. 4. Algorithm for SLUP protocol. Most of notations in this algorithm is based on Definition 4.2, 4.2. For example, l_s denotes the estimated location by state s. The current time is denoted by t, and l_{now} is the very current location determined by a location sensing device such as GPS. The function $computeUITRCost(UWin_i, IWin_i)$ is $\mathcal{UITR}$ cost function of state i performed by Eq. (4).

Table 2
Simulation parameters

Parameters	Meaning	Values Used
N	The number of objects	10,000
$v_{\max}$	Maximum speed	1
δ	Distance threshold	2.0
T	Simulation time	1,000
τ	Temporal locality	0.0 $\sim$ 1.0 (spacing 0.1)
ℓ, ρ	Dominancy	$0 \leqslant \ell, \rho \leqslant \infty$
(ic, tc)	Energy coefficient	(48.61, 71.23)
T_p	Time threshold to switch into pause mode	5
$\hat{T}$	Time threshold of the proposed approach in pause mode	5
$\hat{T}_{time}$	Time threshold of time-based approach	3

generate synthetic datasets which is simulating the realistic movement of real-life objects. We classify the transition matrices for the probability into three types: $\mathbf{T}(\tau)$, $\mathbf{L}(\ell)$, and $\mathbf{S}(\rho)$. These are shown in below:

$$\mathbf{T}(\tau) = \begin{array}{c c} & \begin{array}{cc} \mathsf{L} & \mathsf{R} \end{array} \\ \begin{array}{c} \mathsf{L} \\ \mathsf{R} \end{array} & \begin{pmatrix} \tau & 1-\tau \\ 1-\tau & \tau \end{pmatrix} \end{array}, \text{ where } 0 \leqslant \tau \leqslant 1. \tag{5}$$

$$\mathbf{L}(\ell) = \begin{array}{c c} & \begin{array}{cc} \mathsf{L} & \mathsf{R} \end{array} \\ \begin{array}{c} \mathsf{L} \\ \mathsf{R} \end{array} & \begin{pmatrix} \frac{\ell}{\ell+1} & \frac{1}{\ell+1} \\ \frac{\ell}{\ell+1} & \frac{1}{\ell+1} \end{pmatrix} \end{array}, \text{ where } 0 \leqslant \ell \leqslant \infty. \tag{6}$$

$$\mathbf{S}(\rho) = \begin{array}{c c} & \begin{array}{ccc} \mathsf{P} & \mathsf{L} & \mathsf{R} \end{array} \\ \begin{array}{c} \mathsf{P} \\ \mathsf{L} \\ \mathsf{R} \end{array} & \begin{pmatrix} \frac{\rho}{\rho+1} & \frac{1/2}{\rho+1} & \frac{1/2}{\rho+1} \\ \frac{\rho}{\rho+1} & \frac{1/2}{\rho+1} & \frac{1/2}{\rho+1} \\ \frac{\rho}{\rho+1} & \frac{1/2}{\rho+1} & \frac{1/2}{\rho+1} \end{pmatrix} \end{array}, \text{ where } 0 \leqslant \rho \leqslant \infty. \tag{7}$$

Using these matrices, we can generate realistic workloads in the viewpoint of ℓ, τ, and ρ. The type $\mathbf{T}(\tau)$ has various characteristics with changing τ from 0 to 1 within $\ell = 1$. As τ increases, the generated mobility patterns has a tendency to remain in the current state. The type $\mathbf{L}(\ell)$ has linearity ℓ. The temporal locality of $\mathbf{L}$ matrix is $\sqrt{\ell/(\ell+1)^2}$ by definition, and its maximum value is 0.5. The matrix $\mathbf{L}(0)$ is used to generate the pure random-walk. On the other hand, $\mathbf{L}(\infty)$ is for the extremely linear movement. The type $\mathbf{S}(\rho)$ has various characteristics with changing the stationarity ρ from 0 to ∞. The matrix $\mathbf{S}(\infty)$ can be used to generate stationary objects. 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. Table 2 summarizes setting for the parameters.

5.2. Energy consumption model

In this paper, there are three energy states such as DOZE mode, IDLE mode, and TRANSMIT mode. We now describe the ratio of energy consumption for these states. $\mathcal{E}_s$ describes the amount of energy consumption in an energy state s per unit time.

$$\mathcal{E}_{\mathsf{DOZE}} : \mathcal{E}_{\mathsf{IDLE}} : \mathcal{E}_{\mathsf{TRANSMIT}} = 1 : ic : tc \tag{8}$$

Table 3
The components of Example Mobile Client we examined (in mW) [9]

	Model	doze	normal/active	receive	transmit
CPU	StrongARM SA-1100	0.16	400 (2,500)		
NIC	RangeLAN2 7401/02	25		750 (30)	1,500 (60)
GPS	μ-blox GPS-MS1E	33	462 (14)		

In this paper, the amount of energy consumed in doze mode for unit time is denoted as $unit\ energy$. In many processors, the doze mode has extremely low power consumption. In the Hobbit chip from AT&T, for example, the ratio of power consumption in the active mode to the doze mode is 5,000 [7]. In brief, the "ic" stands for idle coefficient which means the idle-to-doze ratio $\frac{\mathcal{E}_{\mathsf{IDLE}}}{\mathcal{E}_{\mathsf{DOZE}}}$. Similarly, the "$tc$" stands for transmission coefficient which means transmission-to-doze ratio $\frac{\mathcal{E}_{\mathsf{TRANSMIT}}}{\mathcal{E}_{\mathsf{DOZE}}}$.

In this paper, the average energy consumption can be measured by the amount of unit energy in a given time. Undisputedly, this experiment depends sensitively on the pair of (ic, tc). In order to choose reasonable coefficients, we should have some reference values. Table 3 shows the parameters of energy consumption for our experiment [7,9]. From Table 3, we can estimate these coefficients as followings.

$$\begin{aligned}\hat{ic} &= \frac{\text{The total amount of energy consumption in } \mathsf{IDLE} \text{ state}}{\text{The total amount of energy consumption in } \mathsf{DOZE} \text{ state}} \\ &= \frac{(400+750+462)\ \mathrm{mW}}{(0.16+0+33)\ \mathrm{mW}} = 48.61\end{aligned} \tag{9}$$

$$\begin{aligned}\hat{tc} &= \frac{\text{The total amount of energy consumption in } \mathsf{TRANSMIT} \text{ state}}{\text{The total amount of energy consumption in } \mathsf{DOZE} \text{ state}} \\ &= \frac{(400+1,500+462)\ \mathrm{mW}}{(0.16+0+33)\ \mathrm{mW}} = 71.23\end{aligned} \tag{10}$$

5.3. *Effect of dominancy: ℓ and ρ*

In this experiment, we vary the two dominancies (ℓ and ρ) from 0 to ∞ and examine how the proposed approach reduces the energy consumption. Figure 5 shows the average energy consumption in $unit\ energy$, average update cost, and the average imprecision for an object with varying linearity ℓ. For the distance-based approach, increasing ℓ gave rise to increasing the energy consumption. In contrast, dead-reckoning has a better performance than the distance-based approach in the case of higher ℓ. The time-based approach, unlike distance-triggered approaches (distance-based and dead-reckoning), yields constant energy consumption of $\frac{tc+\hat{T}_{time}-1}{\hat{T}_{time}}$. The proposed approach outperforms the traditional ones under all conditions.

In Fig. 6, we consider user's stationarity ρ. When ρ is 0, all moving objects are always moving. As shown in the figure, the energy efficiency has increased as the parameter ρ increased. In the case of $\rho = \infty$, all objects stay at their initial position and the performance of the proposed approach is approximated to $\frac{ic+\hat{T}-1}{\hat{T}}$. Moreover, the performance of distance-triggered and time-triggered approaches are approximated to ic and $\frac{tc+\hat{T}_{time}-1}{\hat{T}_{time}}$ respectively. Such efficiency inherently leads some additional cost of increased location imprecision.

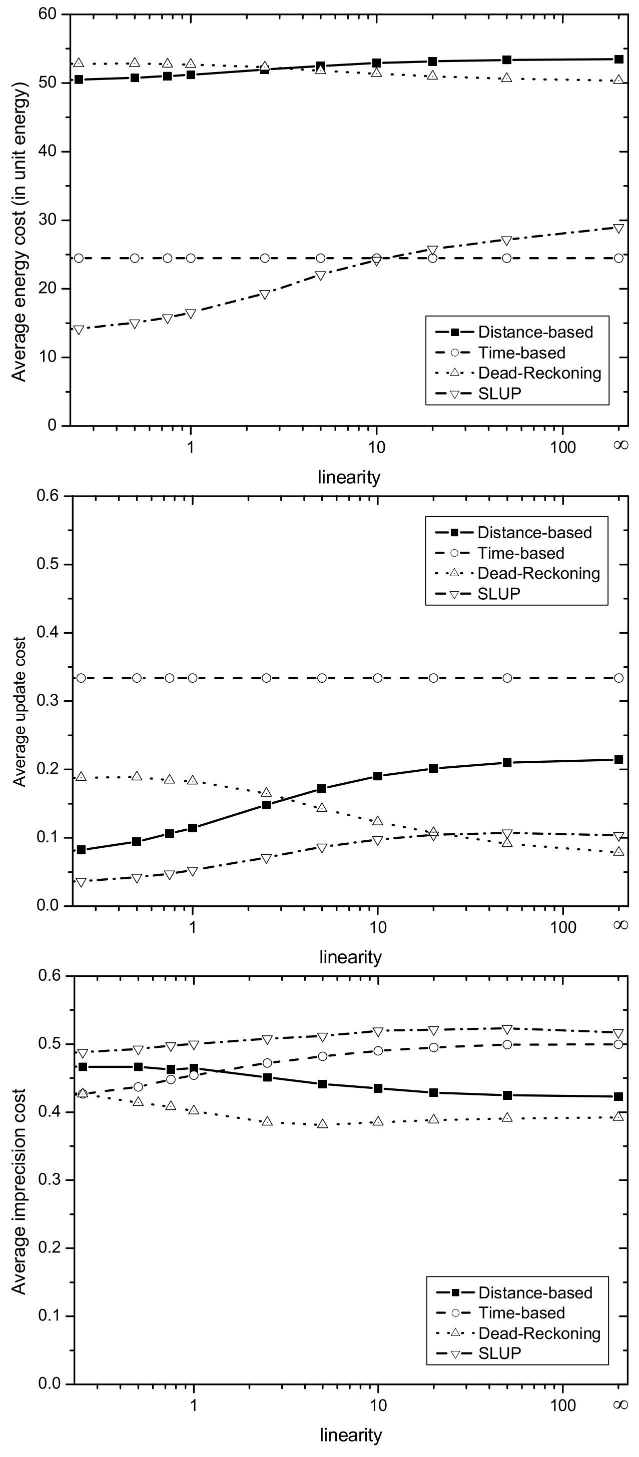


Fig. 5. Effect of linearity ℓ: average energy cost in $unit\ energy$ (first), average update cost (second), and average imprecision cost (third).

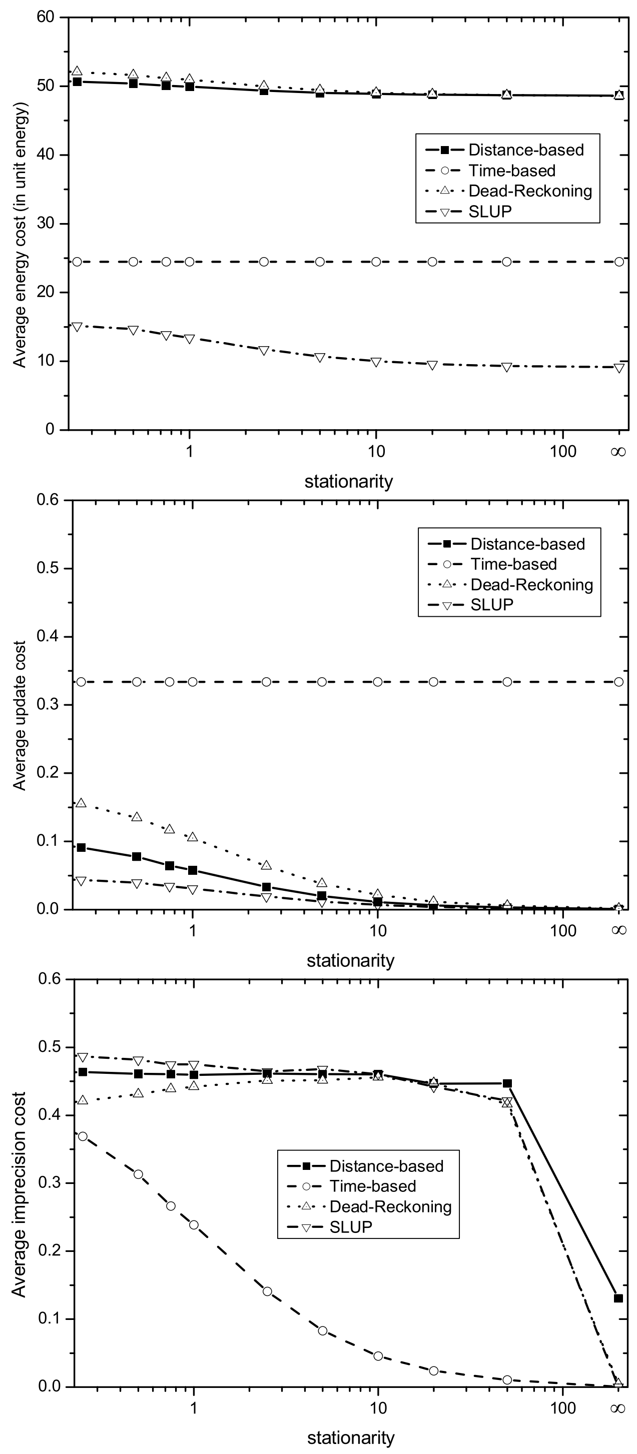


Fig. 6. Effect of stationarity ρ: average energy cost in $unit\ \ energy$ (first), average update cost (second), and average imprecision cost (third).

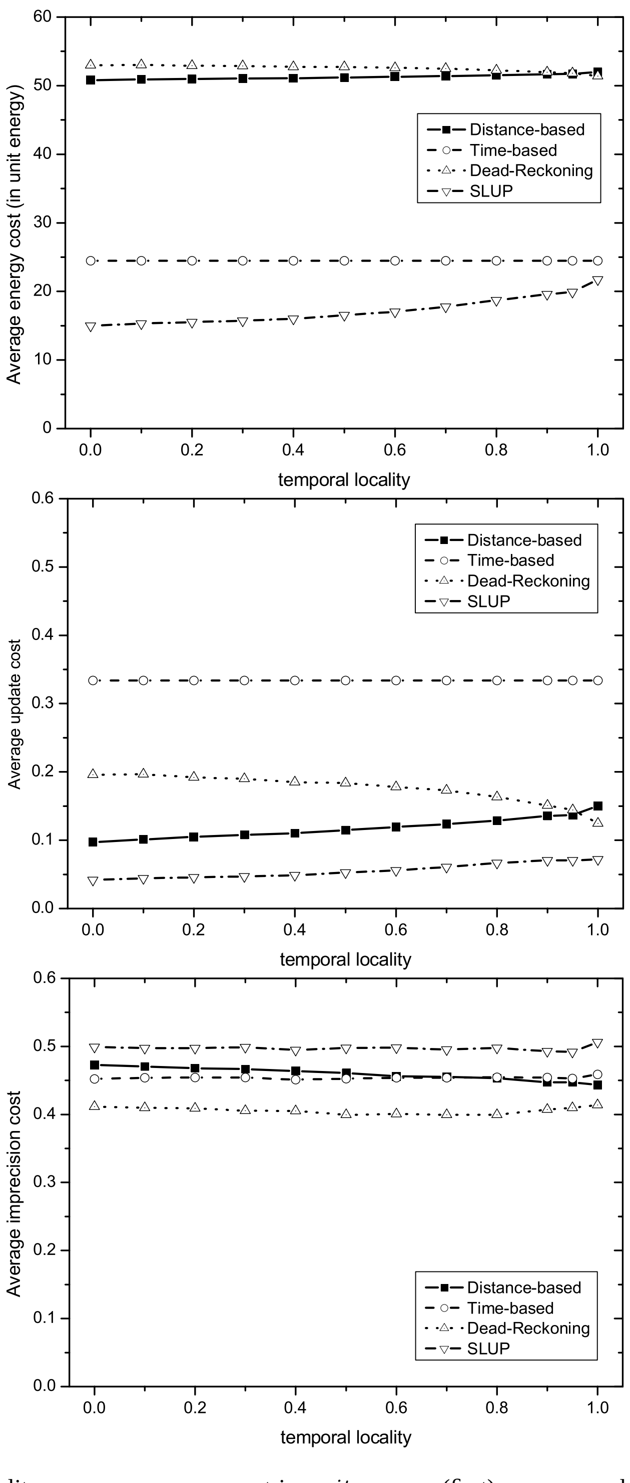
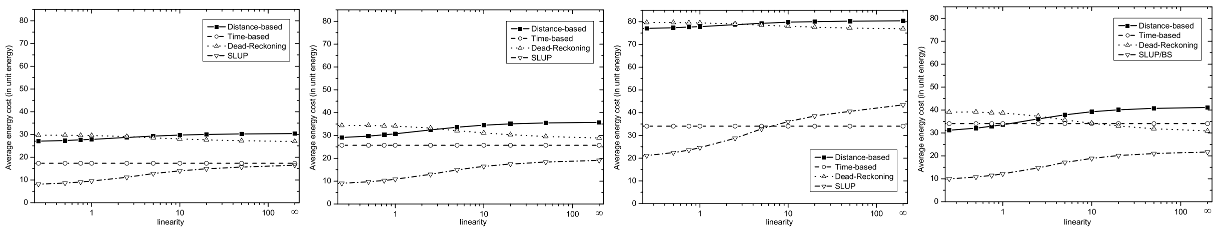
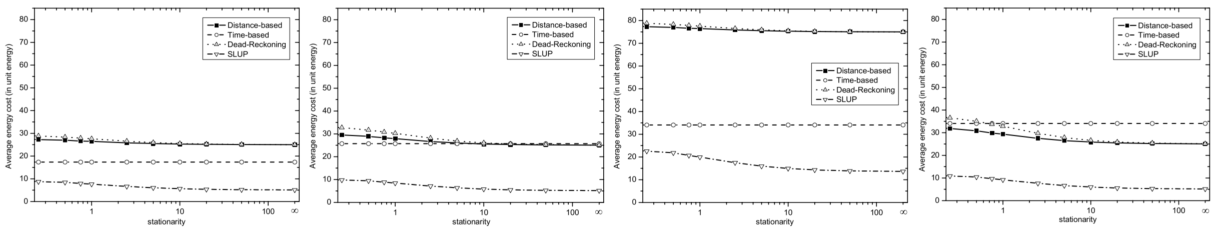


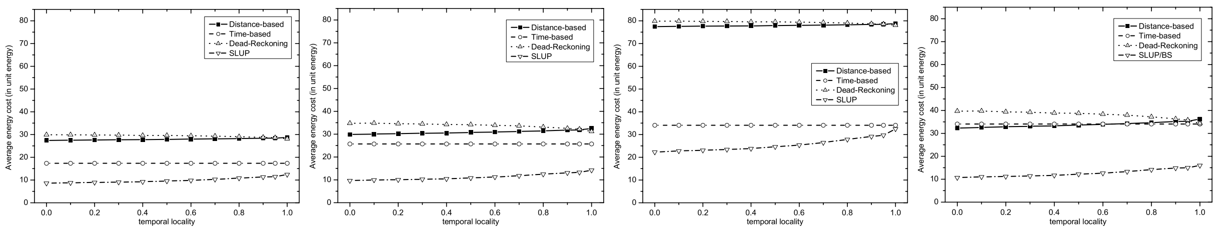
Fig. 7. Effect of temporal locality τ: average energy cost in $unit\ energy$ (first), average update cost (second), and average imprecision cost (third).



(a) Varying linearity l: (25,50), (25,75), (75,100), and (25,100)



(b) Varying stationarity ρ: (25,50), (25,75), (75,100), and (25,100)



(c) Varying temporal locality τ: (25,50), (25,75), (75,100), and (25,100)

Fig. 8. Effect of energy coefficients (ic, tc) in terms of the three mobility parameters, e.g., linearity ℓ, stationarity ρ, and temporal locality τ.

5.4. *Effect of temporal locality* τ

In this experiment, we examine the impact on the temporal locality τ. In Fig. 7, we consider the temporal locality of user movement. Due to fixed linearity 1, a transition matrix with higher τ is likely to have more linear movements than the opposite one. The dead-reckoning approach, therefore, will be advantageous for a higher τ under the same ℓ. The proposed approach outperforms the traditional ones under all condition of varying τ.

5.5. *Effect of energy coefficients* (ic, tc)

In Fig. 8, we consider different energy coefficients such as (25,50), (25,75), (75,100), and (25,100) in terms of the three mobility parameters, e.g., linearity ℓ, stationarity ρ, and temporal locality τ. In every cases the time-triggered (or time-based) approach achieves the same performance, while the distance-triggered approaches (including SLUP) have different performance characteristics in different energy coeffcient (ic, tc). The SLUP protocol outperforms other protocols except in the case of (75,100). Especially in the case of stationarity $\rho = \infty$, the average energy consumption of distance-triggered, time-triggered, and SLUP protocols are ic, $\frac{tc+\hat{T}_{time}-1}{\hat{T}_{time}}$, $\frac{ic+\hat{T}-1}{\hat{T}}$ respectively. Satisfying the inequality

$\frac{ic+\hat{T}-1}{\hat{T}} < ic$ and $\frac{ic+\hat{T}-1}{\hat{T}} < \frac{tc+\hat{T}_{time}-1}{\hat{T}_{time}}$, therefore, the proposed SLUP protocol always outperforms the existing protocols. Solving the inequality $\frac{ic+\hat{T}-1}{\hat{T}} < ic$, we get $1 < ic$. And, solving the inequality $\frac{ic+\hat{T}-1}{\hat{T}} < \frac{tc+\hat{T}_{time}-1}{\hat{T}_{time}}$ (assuming that $\hat{T} = \hat{T}_{time}$), we get $ic < tc$. Consequently, if the inequality of $1 < ic < tc$ is meet, the proposed SLUP protocol always outperforms the existing protocols. As developing mobile devices is trying to minimize the energy consumption in hardware design, we think these energy coefficients will continuously increases.

6. Conclusions

The energy efficiency in mobile computing is considered to be an eternal subject in various layers such as processor, network layer, application layer, and so on. In particular, an energy efficient location tracking mechanism for location-aware computing is desperately needed. In this paper, we have presented a new mobility model called SMM to describe movement patterns of real-life objects in probabilistic manners. Based on this model, we have proposed an energy-efficient location tracking strategy called SLUP. Inherently, this protocol has a tradeoff between accuracy and energy consumption. In our experiment, the additional 9 $\sim$ 10% of imprecision leads to about 80% reduction of energy consumption at maximum.

Future research directions include the generation of spatio-temporal datasets based on our SMM model. We will also extend the proposed approach to dynamically change the distance threshold δ so as to make the best of energy efficiency in location tracking.

References

[1] D. Barbara, Mobile computing and database – A survey, *IEEE Transactions on Knowledge and Data Engineering* **11**(1) (1999).

[2] A. Bhattacharya and S.K. Das, LeZi-update: An information-theoretic framework for personal mobility tracking in PCS networks, *Wireless Networks* **8** (2002).

[3] A. Bar-Noy, I. Kessler and M. Sidi. Mobile Users: To Update or Not To Update?, *Wireless Networks* **1**(2) (1995).

[4] T. Brinkhoff, *Generating Network-Based Moving Objects*, Proc. of SSDBM, July, 2000.

[5] J. Cai and K.-L. Tan, Energy-efficient selective cache invalidation, *Wireless Networks* **5** (1999).

[6] R.H. Guting et al., A foundation for representing and querying moving objects, *ACM Transactions on Database Systems* **25**(1) (March, 2000).

[7] T. Imielinski, S. Viswanathan and B.R. Badrinath, Data on air: Organization and access, *IEEE Transactions on Knowledge and Data Engineering* **9**(3) (1997).

[8] C.E. Jones, K.M. Sivalingam, P. Agrawal and J.C. Chen, A survey of energy efficient network protocols for wireless networks, *Wireless Networks* **7** (2001).

[9] O. Kasten, Energy consumption, ETH-Zurich, Swiss Federal Institute of Technology. Available at http://www.inf.ethz.ch/∼kasten/research/bathtub/energy consumption.html.

[10] D.L. Minh, *Applied Probability Models*, Brooks/Cole Pub., 2001.

[11] E. Pitoura and G. Samaras, Locating objects in mobile computing, *IEEE Transactions on Knowledge and Data Engineering* **13**(4) (2001).

[12] D. Pfoser and Y. Theodoridis. *Generating Semantics-Based Trajectories of Moving Objects*, Int'l Workshop on Emerging Tech. for Geo-Based App., May, 2000.

[13] J.-M. Saglio and J. Moreira, *Oporto: A Realistic Scenario Generator for Moving Objects*, DEXA Workshop on Spatio-Temporal Data Models and Languages, 1999.

[14] M. Song, J. Ryu, S. Lee and C.-S. Hwang, *Considering Mobility Patterns in Moving Objects Database*, Proc. of ICPP, October, 2003.

[15] Y. Theodoridis, J.R.O. Silva and M.A. Nascimento, *On the Generation of Spatiotemporal Datasets*, Proc. of SSTD, 1999.

[16] O. Wolfson, *Moving Objects Information Management: The Database Challenge*, Proc. of NGITS, 2002.
[17] O. Wolfson, A.P. Sistla, S. Chamberlain and Y. Yesha. Updating and querying databases that track mobile units, *Distributed and Parallel Databases* **7**(3) (1999).
[18] 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** (2003).

MoonBae Song received his B.S. degree in computer science from Kunsan National University in 1996, and his M.S. degree in computer science from Soongsil University in 1998. He received his Ph.D. in computer science from Korea University in 2005. He is a researcher in the Research Institute of Computer Information and Communication at Korea University. His research interests include moving object databases, location-aware services, context-awareness, and data management for mobile computing.

Sang-Won Kang received his BS degree in Computer Science and his MS degree in Computer Science and Engineering from Korea University, Seoul, Korea in 1998 and 2003, respectively. He is currently a Ph.D. candidate in Computer Science and Engineering and a researcher in the Research Institute of Computer Information and Communication, Korea University, Korea. His academic interests include Mobile Computing Systems, Mobile Data Management, Location Dependent Data, Semantic Prefetching & caching, Sensor Data Networks, Indexing & Query Processing for Moving Objects.

KwangJin Park received his B.S. and M.S. degree in computer science from Korea University, Korea in 2000 and 2002, respectively. He is currently a Ph.D. candidate in Computer Science and Engineering and a researcher in the Research Institute of Computer Information and Communication, Korea University, Korea. His research interests include location-dependent information systems, mobile databases, and mobile computing systems.