Exploiting Mobility as Context for Energy-efficient Location-aware Computing

MoonBae Song, KwangJin Park and Ki-Sik Kong

Dept. of Computer Science and Engineering, Korea University
5-1, Anam-dong, Seongbuk-Ku, Seoul 136-701, Korea
{mbsong,kjpark,kskong}@disys.korea.ac.kr

Abstract. The "mobility" has two important aspects: i) how to support mobility and ii) how to exploit mobility. This paper considers the latter, while many existing works only consider the former. This work is trying to prove that system performance will be greatly improved by understanding a user's movement. In this paper, we propose a novel location update protocol, called SLUP which minimizes the energy consumption of a mobile client by exploiting a syntactic information of the user's movement. This concept is called *mobility-awareness*. Moreover, there are three variations of the proposed protocol in terms of choosing optimal state: SLUP/BS, SLUP/UITR, and SLUP/IUT$\langle T_{iut} \rangle$. Experimental results show that the proposed protocol outperforms the well-known protocols such as dead-reckoning and distance-based protocol, that the SLUP/IUT$\langle T_{iut} \rangle$ approach can achieve difference performance tradeoffs between energy efficiency and location imprecision by fine-tuning its algorithmic parameter T_{iut}.

1 Introduction

Definition in *Wikipedia* says that "context includes the circumstances and conditions which *surround* an event". Analyzing contextual information of human behavior from low-level sensor data is the most critical issue in context-aware computing.

Recently, the evolution of mobile computing, location sensing, and wireless networking has created a new class of computing: *location-aware computing* (LAC) [3]. A location-aware system is one that knows where each user is located and can use the location-specific information anywhere and anytime. LAC can be viewed as a kind of context-aware computing based on a context of the very "location". While location is only one small part of a user's state, location is a easily-detected and commercially-useful context. In LAC environments, the mobility of a 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. A central problem in LAC is the determination of physical location and the transmission of the location information to a server.

Another major issue in LAC is energy efficiency because of a large population of portable, battery-powered devices. Although the improvement of other

hardware is governed by *Moore's Law*, the lifetime of a battery is expected to increase only 20% over the next 10 years [4]. As a result, attention to power 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 (1) *reducing the number of location update message as possible* and (2) *switching to doze mode as possible*. These two approaches need also awareness of user movement.

However, this movement information used to be hidden by the process of abstraction. As stated in [10], the traditional way of treating mobility is "abstracting-it-away" method. In this method, the mobility information is abstracted in the network layer and hidden from applications in the paradigm of layered abstractions. In [10], M. Grossglauser claims that "if we break the traditional networking abstractions, mobility becomes an opportunity".

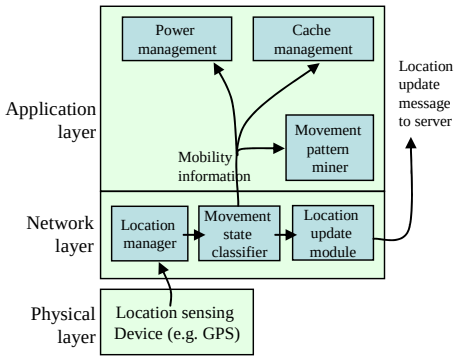


Fig. 1. The Proposed Framework for Mobile Clients in Location-aware Computing

This work is trying to prove that system performance will greatly improved by understanding the user's movement. From the viewpoint of information processing, this issue can help to solve many important issues in LAC: energy efficiency, query processing, caching, indexing, and so forth. Therefore, we can imagine a mobility-aware framework for mobile clients in LAC (Fig. 1). In this figure, the mobility information is not abstracted in the network layer, but forwarded to the application layer in a processed form which is directly exploitable in upper layer applications.

2 Backgrounds

The "location" has two important aspects: i) data that continuously changing and ii) context. In traditional DBMS's, data is assumed to be constant unless it is explicitly modified by an update operation. The former case is location of moving objects that is changing continuously. Thus, in order to represent moving objects (e.g. cars) in a database, the car's location has to be continuously updated. This is unacceptable since either the location is updated very frequently, or the answer to a query is outdated. Furthermore, it is possible that due to disconnections an object cannot continuously update its location. This needs a different approach, called moving objects databases (MOD). In [8], the authors propose a data model called Moving Objects Spatio-Temporal. Its main contribution is the concept of *dynamic attributes*, i.e. attributes that change continuously as a function of time, without being explicitly updated. Research in MOD focuses on ways to

store, index, and retrieve objects and their continuously changing locations in an efficient manner. In the latter case, there are two examples such as location-dependent mobile query (LDMQ) and *location-awareness*. The LDMQ is a query which has different results according to a user's current location. An typical example of the query is "Find a nearby restaurant". Similar to the LDMQ, *location-awareness* means that mobile devices perform different operations based on a user's location. If a user moves into a movie theater, then the mobile device of the user automatically switch into a vibration mode.

The "mobility" also has two important aspects: i) how to support mobility and ii) how to exploit mobility. Supporting the mobility is one of the key problems in mobile computing in order to enable mobile users to communicate anytime and anywhere without the restriction of geographical locations. The key technologies for communicating with mobile users include WLAN, PCS, and Mobile IP. Depending on where it is working on, the "mobility" could have various forms: component mobility, terminal mobility, user mobility, session mobility, role mobility, and application mobility. As a relatively new field of research, the latter is to empirically investigate various benefits by exploiting a user's movement behavior or pattern.

3 Approach

3.1 Basic Design Principles

Basically, the energy efficiency in mobile computing environments can be done by the following two principles.

- *Reducing the Number of Location Update Messages*: 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 [4]. 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[1]. Reducing the imprecision is, therefore, reducing the number of update messages as possible.
- *Switching to doze mode*: The simplest movement is "don't move"; we called the 'pause' mode. 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. By exploiting the 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 whether an MC switch-off itself or not? For this purpose, an MC should be aware of its movement.

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

3.2 Initial Experiment and Mobility-Awareness

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*). $C_{\mathcal{UITR}}(\Delta, w_u, w_\epsilon) = \frac{\sum_{k=1}^{\Delta}(w_u u_k + w_\epsilon \frac{\epsilon_k}{\delta})}{\Delta}$. 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 $w_u \gg w_\epsilon$. Based on this cost model, we can identify the behavioral difference of existing location update policies. Fig. 2 shows the comparison of distance-based approach and dead-reckoning 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.

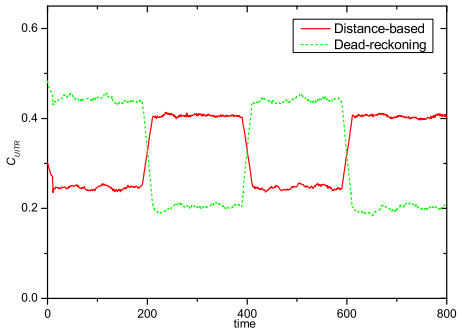


Fig. 2. An example variations of $C_{\mathcal{UITR}}(5, 1, 0.1)$ for two distance-triggered approaches ($\delta = 1.0$)

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). As shown in Fig. 2, a trajectory can be interpreted differently depending on various location update protocols. These interpretations are highly depends on their assumed mobility patterns. The tightness between the performance of location update protocols and their assumed mobility patterns is depicted in the figure. The efficient operations of location update protocols can be achieved by understanding user's movements. Eventually, an MC should be aware of its own mobility. This is what we called *mobility-awareness*.

Based on the initial experiment, therefore, we attempt to classify the movement pattern into a set of small movement components in the viewpoint of location update policy. This serves as a foundation of mobility-awareness. Consequently, the movement pattern is classified into *pause*, *linear movement*, and *random movement*. This classification is ambiguous/relative and may cause a different result from the different location uncertainty. In this paper, we only consider the performance viewpoint of location update policy. In other words,

the syntactic information[2] of current movement state is indirectly recognized by comparing the performance of each location update policy. The concept of *mobility-awareness* is, therefore, a process of inferring the syntactic characteristics of a movement pattern relying on the performance of location update policy.

4 The Proposed Protocol

4.1 Protocol Description

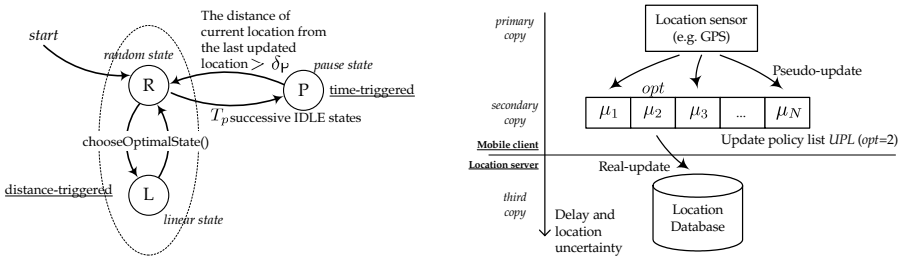
After power-up, each moving object makes transitions through various states depending on its mobility patterns, and updates its location information in location databases using the current update policy (Fig. 3(a)). Because of having more than one update policy, the proposed protocol is called state-based location update protocol (SLUP). The proposed SLUP protocol is represented by a finite set of update policy called *update policy list* $\mathcal{UPL} = \{\mu_1, \mu_2, \ldots, \mu_N\}$ and the optimal update policy index *opt*. Each update policy μ 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. 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. For this purpose, each moving object performs not only the current *opt* update policy but also the others by the pseudo-update procedure which is a self-initiated update procedure within MC's local memory (see Fig. 3(b)). 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.

The behavior of a moving object is modeled as a state-transition diagram. Exploiting temporal locality of mobility patterns, the update policy phase is decomposed into small fraction of *update state* such as P, L, and R. 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.

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[3], then call *chooseOptimalState*() to choose a new (predicted) optimal state and transmit location update message. Otherwise,

[2] In our work, the syntactic information is one of three movement states $\{\mathsf{P}, \mathsf{L}, \mathsf{R}\}$. Similar to our work, Patterson et al. [6] presented a method to learning how a person uses different kinds of transportation in the community. In this work, the mode of transportation is represented as one of three different values $\{\mathsf{BUS}, \mathsf{FOOT}, \mathsf{CAR}\}$.

[3] A state is called *stale* (or inconsistent) if the location information of a state has more location imprecision than a predefined uncertainty δ.



(a) The state transition diagram (b) The flow of location update

Fig. 3. The Proposed Protocol

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.

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 [9]. 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. 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 Fig. 4, behavioral differences of each device in terms of various energy states is presented.

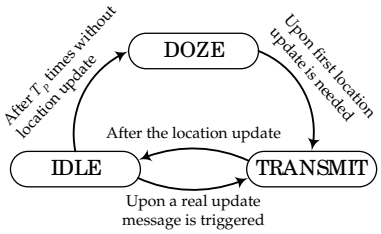


Fig. 4. Energy state transition

4.2 Variations: Choosing the Optimal State

The performance of proposed scheme is greatly dependent on determining the optimal state. When an location update message is sent, a MC should determine whether the location update needs a state transition or not. In order to determine the next *opt* state, *chooseOptimalState*() is called. And there are several variations on the subject.

In Fig. 5, we show three variations. Firstly, one of most simplest approach is "blind switching" (SLUP/BS) in which state transition is required to be done for every location update. With the location update message m_1 in Fig. 5, state transition from L to R is performed concurrently and unconditionally. This is simple, but it is more likely to make a false state transition. The second variation is called SLUP/UITR in which a state with lower $\mathcal{UITR}$ cost among states L and R is selected. In Fig. 5, assuming that the state R is more optimal one than the state L, a state transition is not required to issuing the message m_2. The third

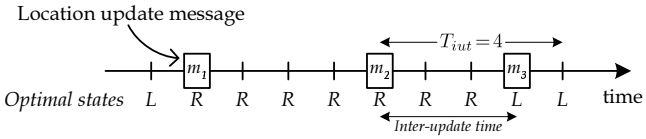


Fig. 5. State Transition Problem: To make or not to make the state transition?

approach considers not a single location update, but a time interval between two consecutive location updates. Let "inter-update time" be a time interval between two consecutive location updates. A state transition will be delayed until the time interval is smaller than a predefined threshold T_{iut}. In Fig. 5, state transition is made with the message m_3, because the time interval between m_2 and m_3 is smaller than $T_{iut}(4)$. This approach is called SLUP/IUT$\langle T_{iut} \rangle$. Obviously, its performance highly depends on T_{iut}. Consequently, the variants of SLUP protocol are choices on the diversity in the degree of laziness for state transition. Such a "laziness" to the location update is very helpful to the energy efficiency.

5 Performance Evaluation

5.1 Simulation Model and Workload Generation

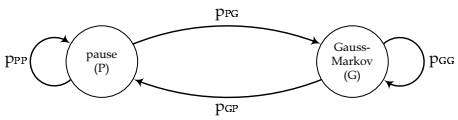
We evaluate the performance of three traditional approach, and four proposed approaches with variation. The traditional approaches include distance-based and dead-reckoning. The proposed approaches include SLUP/BS, SLUP/UITR, and SLUP/IUT$\langle T_{iut} \rangle$ with different T_{iut} (2,5).

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. In this experiment, we adopt the Gauss-Markov mobility model [11]. Specifically, the speed and direction at the n-th instance is calculated based upon the value of speed and direction at the $(n-1)$-th instance and a random variable using the following equations: $s_n = \alpha s_{n-1} + (1-\alpha)\bar{s} + \sqrt{1-\alpha^2} s_{x_{n-1}}$ and $d_n = \alpha d_{n-1} + (1-\alpha)\bar{d} + \sqrt{1-\alpha^2} d_{x_{n-1}}$, where s_n and d_n are new speed and direction of mobile node at time interval n; $0 \leq \alpha \leq 1$, is the tuning parameter used to vary the randomness; and $s_{x_{n-1}}$ and $d_{x_{n-1}}$ are random variables from a Gaussian distribution. By setting $\alpha = 0$ we get a very random motion, while $\alpha = 1$ gives a completely linear motion. Intermediate levels of randomness are obtained by varying the value of α between 0 and 1. In our experiment, mean speed $\bar{s}$ is 1, mean direction $\bar{d}$ is randomly selected from $[0, 2\pi]$, and α is 0.9.

In order to generate more realistic movements, we also consider stationary state, called pause (P). In conjunction with Gauss-Markov model (denoted as G), these are constructed as a form of Markov chain (see Fig. 6(b)). Then, for the purpose of generating synthetic movements, we construct two types of transition matrices: $\mathbf{T}(\tau)$ and $\mathbf{P}(\rho)$. These are shown in below:

Parameters	Values Used
δ (distance threshold)	2.0
τ (temporal locality)	$0.0 \sim 1.0$
ρ (dominancy)	$0 \leq \rho \leq \infty$
$(T_p, \hat{T})$	(5,5)

(a) Simulation Parameters



(b) State Diagram for Workload Generation

Fig. 6. Simulation Parameters and State Diagram for Workload Generation

$$\mathbf{T}(\tau) = \begin{array}{c} \\ \mathsf{P} \\ \mathsf{G} \end{array} \begin{array}{c} \begin{array}{cc} \mathsf{P} & \mathsf{G} \end{array} \\ \begin{pmatrix} \tau & 1-\tau \\ 1-\tau & \tau \end{pmatrix} \end{array}, \quad \mathbf{P}(\rho) = \begin{array}{c} \\ \mathsf{P} \\ \mathsf{G} \end{array} \begin{array}{c} \begin{array}{cc} \mathsf{P} & \mathsf{G} \end{array} \\ \begin{pmatrix} \frac{\rho}{\rho+1} & \frac{1}{\rho+1} \\ \frac{\rho}{\rho+1} & \frac{1}{\rho+1} \end{pmatrix} \end{array} \tag{1}$$

The matrix $\mathbf{T}(\tau)$ is for observing the variation of temporal locality. The temporal locality is denoted as τ which is defined as $\sqrt{p_{\mathsf{PP}} p_{\mathsf{GG}}}$. And the matrix $\mathbf{P}(\rho)$ represents how much the state P has an influence on the whole movement pattern. This measurement is called stationarity ρ, which is defined as $\frac{\sum p_{i\mathsf{P}}}{\sum p_{i\mathsf{G}}}$. Fig. 6(a) 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 *transmission* 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. $\mathcal{E}_{\mathsf{DOZE}} : \mathcal{E}_{\mathsf{IDLE}} : \mathcal{E}_{\mathsf{TRANSMIT}} = 1 : ic : tc$. In this paper, the amount of energy consumed in doze mode for unit time is denoted as *unit energy* which is 33.16 mW in our experiment. 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 [4]. 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 1 shows the parameters of energy consumption for our experiment [5]. Then, the energy coefficients can estimated. $\widehat{ic} = \frac{(400+750+462)\ mW}{(0.16+0+33)\ mW} = 48.61$, and $\widehat{tc} = \frac{(400+1{,}500+462)\ mW}{(0.16+0+33)\ mW} = 71.23$. Thus, the parameters ic and tc is fixed with $(48.61, 71.23)$.

Table 1. The components of Example Mobile Client we examined (in mW) [5]

	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)		

5.3 Effect of stationarity ρ and temporal locality τ

In this experiment, we vary stationarity ρ and temporal locality τ for observing the variety of moving behavior of mobile clients. In Fig. 7, we consider user's stationarity ρ from 0 to ∞ and location uncertainty δ is 2.0. When ρ is 0, all moving objects always moving. If ρ is ∞, all MCs stay at their initial position. As shown in the figure, the energy efficiency has increased as the parameter ρ increased. In the case of $\rho = \infty$, all objects is always on the move and the performance of the proposed approach is approximated to $\frac{ic+\hat{T}-1}{\hat{T}}$. Moreover, the performance of distance-triggered approaches is approximated to ic. Such efficiency inherently leads some additional cost of increased location imprecision. However this overhead is quite smaller than predefined uncertainty δ.

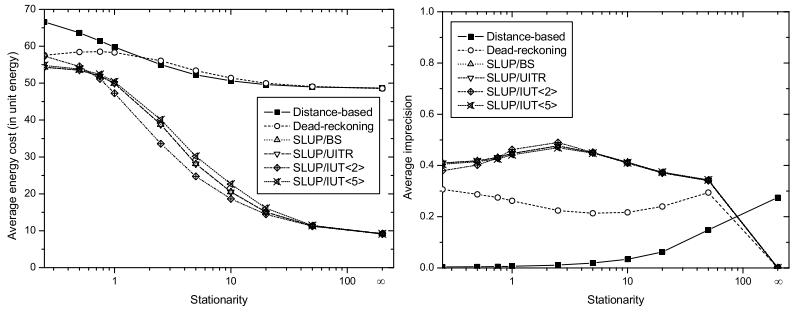


Fig. 7. Effect of stationarity ρ: energy cost and imprecision cost

In Fig. 8, we examine the impact on the temporal locality τ of user movement. Due to fixed $\rho = 1$, the transition matrix with higher τ is likely to have more stationary or linear movements than the opposite one. The dead-reckoning approach, therefore, will be outperform distance-based approach for a higher τ.

6 Conclusions

Context-awareness has motivated various research topics. As a branch of the concept, this work is trying to prove that system performance will greatly improved by understanding the user's movement which is called *mobility-awareness* in the paper. On the basis of our experiment, we argue that the proposed concept,

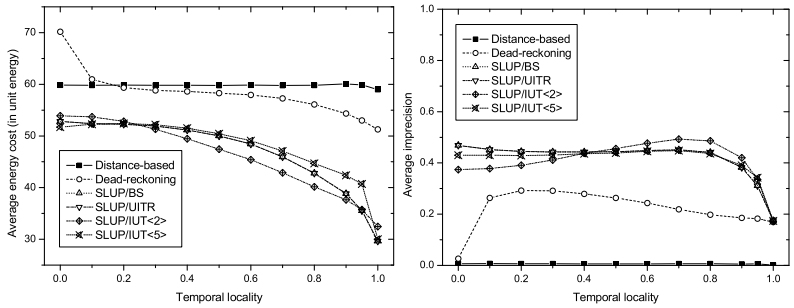


Fig. 8. Effect of locality τ: energy cost and imprecision cost

mobility-awareness, is the key to reducing the energy consumption of mobile clients. This paper also presents a novel location update protocol, called SLUP which minimizes the energy consumption of a mobile client by exploiting a syntactic information of the user's movement. Moreover, there are three variations of proposed protocol in terms of choosing optimal state: SLUP/BS, SLUP/UITR, and SLUP/IUT$\langle T_{iut} \rangle$.

References

1. A. Bhattacharya and S. K. Das. "LeZi-update: An information-theoretic framework for personal mobility tracking in PCS networks," *Wireless Networks*, 8, 2002.
2. A. Bar-Noy, I. Kessler, and M. Sidi. "Mobile Users: To Update or Not To Update?," *Wireless Networks*, 1(2), 1995.
3. Computer Science and Telecommunications Board (CSTB), National Research Council. *IT Roadmap to a Geospatial Future*, The National Academies Press, 2003.
4. T. Imielinski, S. Viswanathan, and B.R. Badrinath. "Data on Air: Organization and Access", *IEEE Trans. on Knowledge and Data Engineering*, 9(3), 1997.
5. O. Kasten, Energy consumption. Available at `http://www.inf.ethz.ch/~kasten/research/bathtub/energy_consumption.html`.
6. Donald J. Patterson, Lin Liao, Dieter Fox, and Henry A. Kautz. "Inferring High-Level Behavior from Low-Level Sensors", *Proc. of Ubicomp* 2003.
7. MoonBae Song, JeHyok Ryu, SangKeun Lee, and Chong-Sun Hwang. "Considering Mobility Patterns in Moving Objects Database," *Proc. of ICPP*, October 2003.
8. 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.
9. Y.-C. Tseng, et al. "A Stop-or-Move mobility model for PCS networks and its location-tracking strategies," *Computer Communications*, 26:1288–1301, 2003.
10. Matthias Grossglauser, "Mobility in Ad Hoc Networks: Friend or Foe?", *Self-organized Wireless Communication Systems*, I&C Research Day 2004.
11. B. Liang and Z.J. Haas, "Predictive distance-based mobility management for PCS networks", In *Proc. of INFOCOM*, 1999.