Mobility Profiling Using Markov Chains for Tree-Based Object Tracking in Wireless Sensor Networks

Li-Hsing Yen Chia-Cheng Yang
Dept. Computer Science & Information Engineering
Chung Hua University
Hsinchu, Taiwan 300, R.O.C.
lhyen@chu.edu.tw

Abstract

Object tracking in wireless sensor networks is to track moving objects by scattered sensors. These sensors are typically organized into a tree to deliver report messages upon detecting object's move. Existing tree construction algorithms all require a mobility profile that is obtained based on historical statistics. In this paper, we propose an analytic estimate of such mobility profile based on Markov-chain model. This estimate replaces otherwise experimental process that collects statistical data. Simulation results confirm the effectiveness of the proposed approach.

1 Introduction

Rapid progress in wireless communications and micro-sensing MEMS technology has enabled the deployment of wireless sensor networks. A wireless sensor network (WSN) consists of a large number of sensor nodes deployed in a region of interest. Each sensor node is capable of collecting, storing, and processing environmental information, and communicating with other sensors.

Object tracking is an application of WSNs where the presence of particular objects (animals, vehicles, etc.) can be detected by nearby sensors. When such event occurs, the location status of the object is reported back to a remote node called *sink*. Specifically, a sensor sends a *join* message to the sink node when it detects the target object entering into its observation area. Similarly, a *leave* message is sent to the sink node when the target is observed out of some sensor's observation. These messages constantly update the target's location information stored in the sink node, where end users can then track the target.

The report of location status would consume a considerable amount of energy if report messages are transmitted directly to the sink node. As a remedy, sensor nodes are organized into a tree [1, 9] rooted at the sink node. A direct transmission from some sensor to the sink node is then replaced by a series of short-range communications, each corresponds to a hop along the unique path form the sensor to the sink node. This technique saves power as radio signal attenuates nonlinearly with distance [3].

The notion of in-node processing further reduces power consumption. The basic idea is not to always propagate report messages all the way to the sink node. Instead, a report message is delivered hop by hop to some sensor node where the current message propagation path joins the previous. Consider the primary scenario where an object moves from one sensor's observation area into another. This event causes two messages to be transmitted, one *leave* and the other *join*. It is uncertain which one will arrive at the sink node first. In fact, the nearest common ancestor of these two sensors terminates the remaining delivery of the update report that arrives secondly. Such a message-pruning technique conserves energy as fewer hops are conducted. The negative effect is that the up-to-date location status may not be available at the sink node, so end users need to issue a query message to retrieve the latest status. The query message is propagated from the sink node to the sensor where the last report was received and stored.

To quantify the benefits of different tree-structuring methods with the effects of in-node processing, we may estimate the delivery cost of an update report to be the number of hops such a delivery takes. Kung et al. [5] have proposed a greedy tree-structuring algorithm DAB, which works in a bottom-up fashion. Their work, however, assumes that all tracking sensors are

the leaves of the tree. Non-leaf nodes have the ability of relaying update reports but not detecting targets. Lin et al. [7] remove such a limitation. They observed that two neighboring sensors with high object crossing rates should be assigned as parent and child, respectively.

All existing tree-structuring algorithms require a mobility profile as an input that describes object crossing rate between every pair of neighboring sensors. Such a profile can be obtained based on historical statistics. However, historical data for an unexplored region may not be available. In this paper, we present a mathematical model that generates a mobility profile by the theory of stochastic process. This modeling is useful especially when the mobility pattern of target objects is unknown. We have conducted computer simulations, where two commonly-adopted mobility models, Random Waypoint [4, 2] and Gauss-Markov [6], were used to drive the movement of a target object. The results confirm the effectiveness of the proposed model. Another contribution of this paper is the proposal of a simple tree-construction algorithm. With the use of the analytic mobility profile, this algorithm can perform better than previous work in the amount of report messages.

The remainder of this paper is organized as follows. The next section states the problem and Section 3 details our analytic work. Experimental results are presented in Sec. 4. The last section concludes this paper.

2 Problem Statement and Related Work

We assume a WSN consisting of N sensors placed in a closed region. The location of every sensor need not be engineered or predetermined but should be known after sensor deployment. A sensor can detect and report an object that is within some range from it. To avoid redundant reports, we assume that only the sensor the is closest to the target object is in charge of the reporting. We also assume that the whole deployment region is fully covered, so the duty area of every sensor can be obtained by drawing a Voronoi diagram on the deployment region (See Fig. 1). The Delaunay triangulation associated with the Voronoi diagram is a graph $G(V, E)$ where V is the set of all sensors and edge $e(i, j) \in E$ for all $i, j \in V$ if i and j share a common border in the Voronoi diagram. The graph can be used to represent a mobility profile by labeling each edge $e(i, j)$ with a weight $w_{i,j}$ that represents the object crossing rate between sensors i and j. Fig. 2(a) shows an example.

The mobility profile has been required by existing

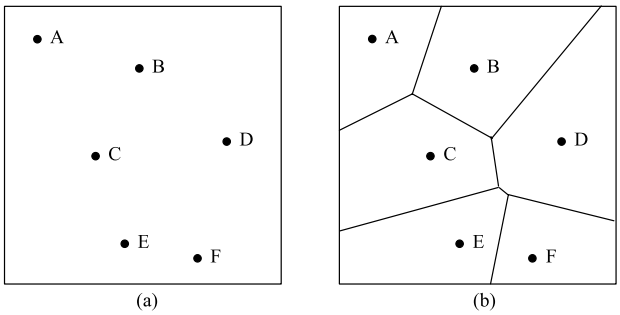

Figure 1. (a) Some sensor deployment and (b) the corresponding Voronoi diagram.

tree construction algorithms. In DAB [5], every sensor i has a weight w_i that is defined by $w_i = \max_j\{w_{i,j}\}$, and sensors are considered adding into a tree in a decreasing order of their weights. In DAT [7], edges in the mobility profile are considered adding into a tree in a decreasing order of their weights. These algorithms do not work if associated weights are absent.

Conventionally, the values of $w_{i,j}$'s are obtained by simulating an object moving around the deployment field. The mobility pattern of the object thus has a significant impact on the resulting weights. How long the object should move to yield an archetypal statistic is another issue. Given a deployment of a set of sensors, our mission is to estimate object crossing rates without actually simulating object movements. This shall be done in the next section.

Both DAB and DAT are heuristic approaches, though it was reported that DAT performs better than DAB [7]. DAT derives a tree from a given graph that represents a mobility profile. The resultant tree possesses the property of "deviation-free"—the path from every node to the sink along the tree must be one of those in the graph that have the minimum hop count. The authors claimed that the superiority of DAT over DAB comes from the deviation-free property. However, we have observed that this property may not be a plus when weight distribution is non-uniform. For example, for the mobility profile shown in Fig. 2(a), DAT will derive a deviation-free tree as shown in Fig. 2(b). This tree is not good enough considering the fact that the highest object crossing rate lies between C and A, which should be connected in the tree to minimize report message transmissions. Fig. 3 shows a better tree that can be derived from the same mobility profile. This tree cannot be generated by DAT since it is not deviation-free: the minimum hop count from B to the sink is 2 in the graph while the actual hop count in the tree is 3.

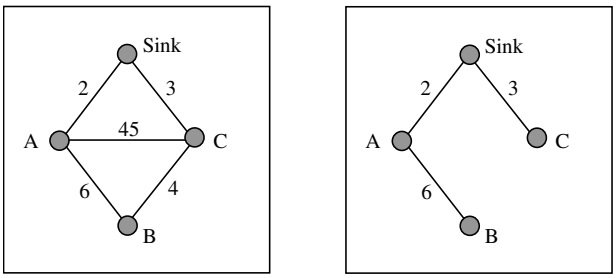

Figure 2. (a) A mobility profile. (b) The tree derived from (a) by DAT.

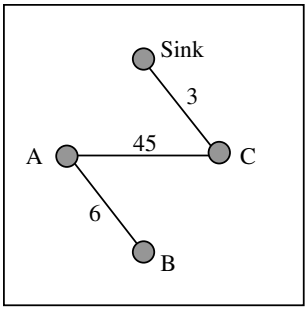

Figure 3. A better tree derived from Fig. 2(a).

As an alternative, we propose a tree construction algorithm called Maximum Spanning Tree (MST), which derives a spanning tree from a given graph that has the maximum total weight. In fact, MST derives the tree shown in Fig. 3 given the same mobility profile. MST performs better than DAT when edge weights vary significantly. The general performance of MST depends on the mobility pattern of objects. We shall explore this issue by simulations in Sec. 4.

3 Proposed Estimate

The basic idea behind our approach is to model object movements as a stochastic process $X(t)$. Suppose, without loss of generality, that all sensors are numbered from 1 to N. When an object is observed by sensor i at time t, we say that the object is in state i at time t, denoted by $X(t) = i$. To simplify the model, we assume that time values are all discrete and advance to the next value only when state changes.

For any integer $n \geq 0$ and $i \in \{1, \ldots, N\}$, let $p_i^n = \Pr[X(n) = i]$ be the probability that the process is in state i at time n and define $\mathbf{p}(n)$ to be the n-th state vector given by $\mathbf{p}(n) = \langle p_1^n, p_2^n, \cdots, p_N^n \rangle$. Since we have no prior knowledge of object's appearance, it is reasonable to assume that the probability of the target object being in state i initially is proportional to the size of sensor i's duty area. That is,

$$p_i^0 = \frac{A_i}{\sum_N A_j},$$

where A_i is the size of sensor i's duty area.

To resolve our problem, state transition probabilities for this process must be obtained. Let $\mathbf{M}(m, n)$ be the transition probability matrix for times m and n, where $m < n$. Specifically,

$$\mathbf{M}(m,n) = \begin{pmatrix} p_{1,1}^{m,n} & p_{1,2}^{m,n} & \cdots & p_{1,N}^{m,n} \\ p_{2,1}^{m,n} & p_{2,2}^{m,n} & \cdots & p_{2,N}^{m,n} \\ \vdots & \vdots & \cdots & \vdots \\ p_{N,1}^{m,n} & p_{N,2}^{m,n} & \cdots & p_{N,N}^{m,n} \end{pmatrix},$$

where

$$p_{i,j}^{m,n} = \Pr[X(n) = j | X(m) = i].$$

By definition, we have $\mathbf{p}(n) = \mathbf{p}(m) \times \mathbf{M}(m, n)$.

If this process satisfies the Markov property, i.e., the conditional probability that $X(n) = j$ given $X(m) = i$, $X(m+1) = i_{m+1}$, ..., $X(n-1) = i_{n-1}$ depends only on the most recent observation $X(n-1) = i_{n-1}$, not on other history, then the state transition probabilities can be easily decomposed by applying the Chapman-Kolmogorov Equation

$$p_{i,j}^{m,n} = \sum_{k=1}^{N} \left(p_{i,k}^{m,r} \times p_{k,j}^{r,n} \right)$$

for any time $m < r < n$. Unfortunately, the Markov property may not hold in some mobility models. Mobility model like Random Waypoint [4, 2] tends to maintain the current moving direction when a object crosses a border of sensor's duty areas. Therefore, if an object has changed states from i to j, the object is more likely to enter the duty area pointed to by the straight line connecting i to j than to any other. Fig. 4 shows an example.

Nevertheless, we model the object movement process as a Markov chain (which possesses the Markov property). Consequently, we have

$$\mathbf{p}(n) = \mathbf{p}(0) \times [\mathbf{M}(0,1) \times \mathbf{M}(1,2) \times \cdots \times \mathbf{M}(n-1,n)]. \quad (1)$$

We assume that sensor nodes never move after deployment and operate long enough to accomplish their tracking mission. This assumption makes the chain a stationary process, which implies $\mathbf{M}(0,1) = \mathbf{M}(1,2) = \cdots = \mathbf{M}(n-1,n)$. Eq. (1) therefore becomes

$$\mathbf{p}(n) = \mathbf{p}(0) \times \mathbf{M}^n, \quad (2)$$

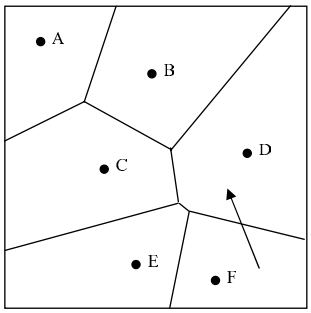

Figure 4. An object has moved from F's duty area to D's. The object is more likely to enter B's duty area than to either C's, E's, or F's.

where $\mathbf{M} = \mathbf{M}(0,1)$ is a one-step transition probability matrix. Specifically,

$$\mathbf{M} = \begin{pmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,N} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,N} \\ \vdots & \vdots & \cdots & \vdots \\ p_{N,1} & p_{N,2} & \cdots & p_{N,N} \end{pmatrix},$$

where $p_{i,j} = \Pr[X(t+1) = j | X(t) = i]$ for any $t \geq 0$. The values of $p_{i,j}$'s can be computed as follows. Any sensor shares common borders with its neighbors, one for each. Suppose sensor i has k neighbors with common border lengths $l_1, l_2, \ldots, l_k$, respectively. The probability that an object moves to the duty area of the j-th neighbor ($1 \leq j \leq k$), given the fact that it is current under sensor i's surveillance, is $l_j/(l_1 + l_2 + \cdots + l_k)$.

It is not difficult to see that $\mathbf{M}^n$ converges as $n \to \infty$. First, the Markov chain is *irreducible* as an object starting at any state has a non-zero probability to eventually visit any other state. Second, the process is *recurrent* since the number of times any state is entered is infinite if $n \to \infty$. It is *positive recurrent* because the expected return time (i.e., the time between any two consecutive visits to the same state) is finite. Third, the process is *aperiodic* since the return time is not fixed for any state. It follows from these properties [8] that the Markov chain has a limiting state probability $\pi = \langle \pi_1, \pi_2, \ldots, \pi_N \rangle = \lim_{n\to\infty} \mathbf{p}(n) = \mathbf{p}(0) \times \lim_{n\to\infty} \mathbf{M}^n$. It also follows that

$$\lim_{n\to\infty} \mathbf{M}^n = \begin{pmatrix} \pi \\ \pi \\ \vdots \\ \pi \end{pmatrix}.$$

With the contents of $\mathbf{M}$ and π, the weight associated with each edge (i,j) in the mobility profile can be calculated as

$$w_{i,j} = (\pi_i \times p_{i,j} + \pi_j \times p_{j,i}).$$

Table 1. All possible mobility-profile/tree-construction combinations.

	Tree construction	
Mobility profile	DAT	MST
Statistical	DAT(S) [7]	MST(S)
Analytical	DAT(A)	MST(A)

4 Simulation Results

We conducted simulations to compare the difference between using statistical and analytical mobility profiles. The deployment field was a $100 \times 100\,\mathrm{m}^2$ area. The number of deployed sensors was varied from 10 to 500. Both DAT and MST were tested in the simulations.

Two mobility models were used to drive object's movements, Random Waypoint and Gauss-Markov. In Random Waypoint model, an object is randomly placed initially. It then randomly selects a destination location to move with a randomly determined speed. The object waits a random paused time when it arrives at the destination and then moves to another location following the same random distributions. In Gauss-Markov model, a randomly placed object determines its moving speed and direction randomly. After a fixed period of time (5 sec. in our setting), the object alters the speed and direction according to Gaussian distributions with the current values of speed and direction as their respective means. To preclude extreme values, any randomly generated value outside the range $[\mu - 2\sigma, \mu + 2\sigma]$ was discarded and regenerated, where μ and σ are mean and standard deviation of the distribution, respectively. In our setting, the standard deviations of speed and direction are 2.5 m/sec. and 20 degrees, respectively. We further limit the speed value to the range between 1 and 50 m/sec.

In all experiments, a mobility profile was first generated by running an object around the deployment field according to Random Waypoint or Gauss-Markov mobility model. The object took 150,000 moves in each run to ensure that the resultant mobility profile is statistically significant. In case of using historical statistics, the mobility profile was then used to construct a data delivery tree. In case of using the analytic estimate, the data delivery tree is constructed by using estimated weights. Both DAT and MST were used for

tree constructions, resulting in four possible combinations of resultant trees as listed in Table 1. The cost of a resultant tree was measured by counting the number of message transmissions that took place when rerunning the object according to the statistical mobility profile. Figs. 5 and 6 show the obtained results.

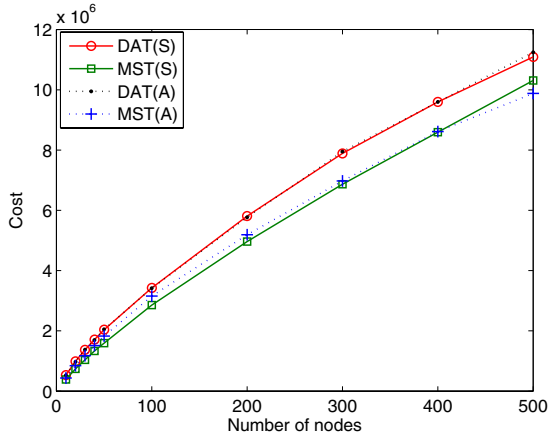

Figure 5. Tree costs with object's movements driven by Random Waypoint model. All values are averaged over 100 runs.

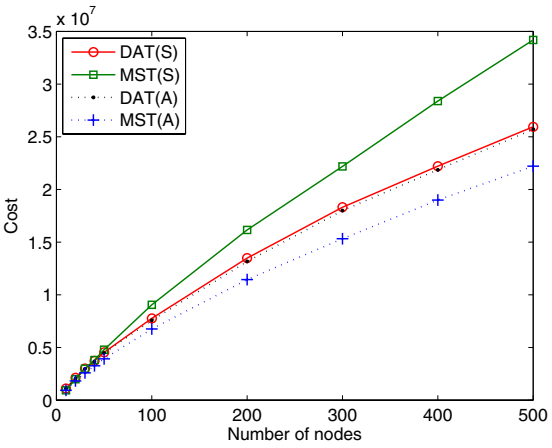

Figure 6. Tree costs with object's movements driven by Gauss-Markov model. All values are averaged over 100 runs.

In both mobility models, DAT using an analytic mobility profile, DAT(A), performs nearly the same as that using a statistical mobility profile, DAT(S). MST performs even better with the analytic estimate than with historical data. This confirms the effectiveness of the proposed estimate.

When Random Waypoint model was used to simulate object's movements, MST has lower cost than DAT has regardless of which type of mobility profiles was used. When object's movements were driven by Gauss-Markov model, the results are twofold. MST with analytic mobility profile (MST(A)) still has lower cost than DATs. However, MST with statistical mobility profile (MST(S)) has the highest cost among all. This can be explained as the depths of trees increase with the number of nodes. The increasing rate is low for DATs and high for MSTs. In particular, MST(S) under Gauss-Markov model has the highest increasing rate. Meanwhile, the maximum weights in mobility profiles tend to decrease with the number of nodes, devaluing the design talent of MST. Consequently, MST(S) performs the worst with a large number of nodes in Gauss-Markov model. The depth increasing rate for MST(A) is lower than that for MST(S). Therefore, MST(A) still performs better than both DAT(S) and DAT(A).

5 Conclusions

We have analyzed object crossing rates between any two sensors deployed in a closed region. The analytical results serve as a mobility profile that is essential to tree-construction algorithms, which derive a data-propagation tree from the mobility profile. A good data-propagation tree can save power on message transmissions, prolonging network lifetime. We have conducted simulations to examine the performance with our analytic work. The results show that an analytical estimate can replace the otherwise historical statistics without any performance degradation. We also have presented another tree-construction algorithm, MST. Experimental results have shown that MST using the proposed analytic mobility profile outperforms previous work.

Acknowledgement

This work has been jointly supported by the National Science Council, Taiwan, under contract NSC-94-2213-E-216-001 and by Chung Hua University under grant CHU-94-TR-02.

References

[1] V. Annamalai, S. K. S. Gupta, and L. Schwiebert. On tree-based convergecasting in wireless sensor networks. In *Proc. IEEE WCNC*, pages 1942–1947, Mar. 2003.

[2] C. Bettstetter. Smooth is better than sharp: A random mobility model for simulation of wireless networks. In *Proc. 4th ACM Int'l Workshop on Modeling, Analysis,*

and Simulation of Wireless and Mobile Systems, pages 19–27, Rome, Italy, July 2001.

[3] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *Proc. 33rd Annual Hawaii Int'l Conf. on System Sciences*, pages 1–10, Jan. 2000.

[4] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. In T. Imieliński and H. F. Korth, editors, *Mobile Computing*, pages 153–181. Kluwer Academic Publishers, 1996.

[5] H. T. Kung and D. Vlah. Efficient location tracking using sensor networks. In *Proc. IEEE WCNC*, pages 1954–1961, Mar. 2003.

[6] B. Liang and Z. J. Haas. Predictive distance-based mobility management for multidimensional PCS networks. *IEEE/ACM Trans. on Networking*, 11:718–732, Oct. 2003.

[7] C.-Y. Lin, W.-C. Peng, and Y.-C. Tseng. Efficient in-network moving object tracking in wireless sensor networks. *IEEE Trans. on Mobile Computing*, to appear.

[8] S. M. Ross. *Stochastic Processes*. John Wiley & Sons, New York, 2nd edition, 1996.

[9] S. Upadhyayula, V. Annamalai, and S. K. S. Gupta. A low-latency and energy-efficient algorithm for convergecast in wireless sensor networks. In *Proc. IEEE GLOBECOM*, pages 3525–3530, San Fransisco, Dec. 2003.