Probability-based Prediction and Sleep Scheduling for Energy Efficient Target Tracking in Sensor Networks

Bo Jiang, *Student Member, IEEE,* Binoy Ravindran, *Senior Member, IEEE,* and Hyeonjoong Cho, *Member, IEEE*

Abstract—A surveillance system, which tracks mobile targets, is one of the most important applications of wireless sensor networks. When nodes operate in a duty cycling mode, tracking performance can be improved if the target motion can be predicted and nodes along the trajectory can be proactively awakened. However, this will negatively influence the energy efficiency and constrain the benefits of duty cycling. In this paper, we present a Probability-based Prediction and Sleep Scheduling protocol (PPSS) to improve energy efficiency of proactive wake-up. We start with designing a target prediction method based on both kinematics and probability. Based on the prediction results, PPSS then precisely selects the nodes to awaken and reduces their active time, so as to enhance energy efficiency with limited tracking performance loss. We evaluated the efficiency of PPSS with both simulation-based and implementation-based experiments. The experimental results show that compared to MCTA algorithm, PPSS improves energy efficiency by $25\% \sim 45\%$ (simulation-based) and 16.9% (implementation-based), only at the expense of an increase of $5\% \sim 15\%$ on the detection delay (simulation-based) and 4.1% on the escape distance percentage (implementation-based) respectively.

Index Terms—Energy efficiency, target prediction, sleep scheduling, target tracking, sensor networks.

1 Introduction

Wireless sensor networks (WSNs) are increasingly being envisioned for collecting data, such as physical or environmental properties, from a geographical region of interest. WSNs are composed of a large number of low cost sensor nodes, which are powered by portable power sources, e.g. batteries [1].

In many surveillance applications of WSNs, tracking a mobile target (e.g., a human being or a vehicle) is one of the main objectives. Unlike detection that studies discrete detection events [2], [3], a target tracking system is often required to ensure continuous monitoring, i.e., there always exist nodes that can detect the target along its trajectory (e.g., with low detection delay [4], [5] or high coverage level [6]). Therefore, the most stringent criterion of target tracking is to track with zero detection delay or 100% coverage.

Since nodes often run on batteries that are generally difficult to be recharged once deployed, energy efficiency is a critical feature of WSNs for the purpose of extending the network lifetime. However, if energy efficiency is enhanced, the quality of service (QoS) of target tracking is highly likely to be negatively influenced. For example, forcing nodes to sleep may result in missing the passing target and lowering the tracking coverage. Therefore, energy efficient target tracking should improve the tradeoff between energy efficiency and tracking performance—e.g., by improving energy efficiency at the expense of a relatively small loss on tracking performance.

For target tracking applications, idle listening is a major source of energy waste [7]. To reduce the energy consumption during idle listening, duty cycling is one of the most commonly used approaches [8]. The idea of duty cycling is to put nodes in the sleep state for most of the time, and only wake them up periodically. In certain cases, the sleep pattern of nodes may also be explicitly scheduled, i.e., forced to sleep or awakened on demand. This is usually called sleep scheduling [9].

As a compensation for tracking performance loss caused by duty cycling and sleep scheduling, proactive wake-up has been studied for awakening nodes proactively to prepare for the approaching target [10], [11]. However, most existing efforts about proactive wake-up simply awaken all the neighbor nodes in the area, where the target is expected to arrive, without any differentiation [6], [10], [12]. In fact, it is sometimes unnecessary to awaken all the neighbor nodes. Based on target prediction [11], [13], [14], it is possible

- *B. Jiang is with Intel Corporation, Hillsboro, OR, 97124. E-mail: jiang.brendan@gmail.com*
- *B. Ravindran is with the Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA, 24061. E-mail: binoy@vt.edu*
- *H. Cho is with the Department of Computer and Information Science, Korea University, Seoul, Korea 136-701. E-mail: raycho@korea.ac.kr*
- *The preliminary result was presented in IPDPS 2008.*

to sleep-schedule nodes precisely, so as to reduce the energy consumption for proactive wake-up. For example, if nodes know the exact route of a target, it will be sufficient to awaken those nodes that cover the route during the time when the target is expected to traverse their sensing areas.

In this paper, we present a probability-based target prediction and sleep scheduling protocol (PPSS) to improve the efficiency of proactive wake-up and enhance the energy efficiency with limited loss on the tracking performance. With a target prediction scheme based on both kinematics rules and theory of probability, PPSS not only predicts a target's next location, but also describes the probabilities with which it moves along all the directions. Unlike other physics-based prediction work [14], target prediction of PPSS provides a directional probability as the foundation of differentiated sleep scheduling in a geographical area. Then, based on the prediction results, PPSS enhances energy efficiency by reducing the number of proactively awakened nodes and controlling their active time in an integrated manner. In addition, we design distributed algorithms for PPSS that can run on individual nodes. This will improve the scalability of PPSS for large-scale WSNs.

Since PPSS depends on kinematics-based target prediction, it primarily aims at tracking a vehicle that usually moves in a smooth curvilinear trajectory without abrupt direction changes.

We evaluated the efficiency of PPSS with both simulation-based and implementation-based experiments. The simulation-based experimental studies show that compared to circle-based proactive wake-up scheme (Circle) [6] and the minimal contour tracking algorithm (MCTA) [12], PPSS introduces an improvement of $25\% \sim 45\%$ on energy efficiency, at the expense of $5\% \sim 15\%$ increase on detection delay. The implementation-based experimental results also show that compared to MCTA, PPSS achieves an improvement of 16.9% on energy efficiency, at the expense of only 4.1% increase on the escape distance percentage.

This paper makes the following contributions:

1) We designed a target prediction scheme based on both kinematics rules and theory of probability, and enhanced the energy efficiency of proactive wake-up with both awakened node reduction and active time control efforts.

2) The proposed distributed algorithms of PPSS, which run on individual nodes, make PPSS scalable for large-scale WSNs.

3) Besides the simulation-based evaluation, we also implemented a prototype on TelosB motes [15] and TinyOS [16] to evaluate PPSS with field experiments. The implementation not only verified the rationality and the feasibility of PPSS, but also strengthened the paper's contributions with more convincing results than those from the simulation.

The rest of the paper is organized as follows. Related work is discussed in Section 2. In Section 3, we introduce system models, our assumptions, and overview the protocol design. We develop the target prediction models in Section 4, then present the energy conservation approaches, including awakened nodes reduction and active time control in Section 5. In Section 6, we specify the distributed algorithms of PPSS. In Section 7, we report the evaluation results from both the simulation and the implementation. We conclude and discuss the future work in Section 8.

2 Related Work

Energy efficiency has been extensively studied either independently or jointly with other features. In [17], the authors proposed, analyzed and evaluated the energy consumption models in WSNs with probabilistic distance distributions to optimize grid size and minimize energy consumption accurately. An experimental effort based on real implementation is conducted for energy conservation in [18]. In [19], Sengul *et al.* explored the energy-latency-reliability tradeoff for broadcast in WSNs by presenting a new protocol called PBBF. In [20], the authors proposed a distributed, scalable and localized multipath search protocol to discover multiple node-disjoint paths between the sink and source nodes, in which energy was considered as a constraint so that the design is feasible for the limited resources of WSNs.

As one of the most important applications of WSNs, target tracking was widely studied from many perspectives. First, tracking was studied as a series of continuous localization operations in many existing efforts [21], [22]. Secondly, target tracking was sometimes considered as a dynamic state estimation problem on the trajectory, and Bayesian estimation methods, e.g., particle filtering, were used to obtain optimal or approximately optimal solutions [23]. Thirdly, in some cases, target tracking was considered as an objective application when corresponding performance metrics, e.g., energy efficiency [6] or real-time feature [4], were the focus. Fourthly, a few efforts were conducted based on real implementation, and emphasized the actual measurement for a tracking application [4]. Finally, a few target tracking efforts did not explicitly distinguish tracking from similar efforts, such as detection [6] and classification [24].

Although sleep scheduling and target tracking have been well studied in the past, only a few efforts [6], [12] investigated them in an integrated manner. In [6], the authors utilize a "circle-based scheme" (Circle) to schedule the sleep pattern of neighbor nodes simply based on their distances from the target. In such a legacy Circle scheme, all the nodes in a circle follow the same sleep pattern, without distinguishing among various directions and distances. In [12], Jeong *et. al.* present MCTA algorithm to enhance energy efficiency

by solely reducing the number of awakened nodes. MCTA depends on kinematics to predict the contour of tracking areas, which are usually much smaller than the circles of Circle scheme. However, MCTA keeps all the nodes in the contour active without any differentiated sleep scheduling.

Typical target prediction methods include kinematics-based prediction [12], [14], dynamics-based prediction [25], and Bayesian estimation methods [23], [26]. Kinematics and dynamics are two branches of the classical mechanics. Kinematics describes the motion of objects without considering the circumstances that cause the motion, while dynamics studies the relationship between the object motion and its causes [27]. In fact, most of past work about target prediction uses kinematics rules as the foundation, even for those that use Bayesian estimation methods.

MCTA algorithm presented in [12] is just an example of kinematics-based prediction. Another example is the Prediction-based Energy Saving scheme (PES) introduced in [14]. It only uses simple models to predict a specific location without considering the detailed moving probabilities.

In [25], Taqi *et al.* discussed a dynamics-based prediction protocol named as A-YAP. They leveraged the physics research results on the yaw rate and the side force. However, these results depend on the target mass, which requires the surveillance system to recognize the target with target classification techniques. In many cases, target classification is difficult especially when the real-time tracking constraint is applied. Moreover, A-YAP also predicts an exact location that the target is probably moving to, instead of considering all the possibilities.

Bayesian estimation methods estimate the target state by incorporating new measures to modify the prior states as well as predict the posterior ones. For example, information-driven sensor querying (IDSQ) [28] optimizes the sensor selection to maximize the information gain while minimizing the communication and resource usage. The enhancement of energy efficiency is not achieved by sleep scheduling, but by minimizing the communication energy. On the contrary, PPSS aims at improving the overall performance on energy efficiency and tracking performance using sleep scheduling.

Another example of Bayesian estimation methods is the particle filtering [23]. In [13], the authors predict the target location using a particle filter, then schedule the sleep patterns of nodes based on the prediction result. Similar to Circle scheme, they schedule the sleep patterns based on the distance only.

3 Design Overview

In this section, we introduce system models, our assumptions, and overview the design of PPSS protocol.

3.1 System Models and Assumptions

We consider a homogeneous, static sensor network, in which sensor nodes work in a duty cycling mode. In each *toggling period* (TP), a node keeps active for $TP * DC$, where DC is the *duty cycle*. Although the active period of neighbor nodes may be different, the communication among them can be guaranteed based on a MAC protocol such as B-MAC [29].

In the active state, a node may detect targets within its sensing radius r, and communicate with other nodes within its communication radius R. We assume that every node is aware of its own location (using GPS [30] or algorithmic strategies such as [31]), and is able to determine a target's position at detection (either by sensing or by calculating—e.g., [10], [24]). In addition, we assume that the sensor nodes are locally time synchronized using a protocol such as RBS [32].

In this paper, we consider single target tracking only. In fact, as long as the distance between two targets is more than two times of the communication radius of nodes, the sleep scheduling actions triggered by them will not overlap, thereby they can be handled with single target tracking algorithms.

3.2 PPSS Design

PPSS is designed based on proactive wake-up: when a node (i.e., *alarm node*) detects a target, it broadcasts an alarm message to proactively awaken its neighbor nodes (i.e., *awakened node*) to prepare for the approaching target. To enhance energy efficiency, we modify this basic proactive wake-up method to sleep-schedule nodes precisely. Specifically, PPSS selects some of the neighbor nodes (i.e., *candidate node*) that are likely to detect the target to awaken. On receiving an alarm message, each candidate may individually make the decision on whether or not to be an awakened node, and if yes, when and how long to wake up.

We utilize two approaches to reduce the energy consumption during this proactive wake-up process:

1) Reduce the number of awakened nodes.
2) Schedule their sleep pattern to shorten the active time.

First, the number of awakened nodes can be reduced significantly, because: 1) those nodes that the target may have already passed during the sleep delay do not need to be awakened; 2) nodes that lie on a direction that the target has a low probability of passing by could be chosen to be awakened with a low probability. For this purpose, we introduce a concept of *awake region* and a mechanism for computing the scope of an awake region.

Secondly, the active time of chosen awakened nodes can be curtailed as much as possible, because they could wake up and keep active only when the target is expected to traverse their sensing area. For this purpose, we present a sleep scheduling protocol, which

schedules the sleep patterns of awakened nodes individually according to their distance and direction away from the current motion state of the target.

Both of these energy reducing approaches are built upon target prediction results. Unlike the existing efforts of target prediction [12], [14], [25], we develop a target prediction model based on both kinematics rules and probability theory. Kinematics-based prediction calculates the expected displacement of the target in a sleep delay, which shows the position and the moving direction that the target is most likely to be in and move along. Based on this expected displacement, probability-based prediction establishes probabilistic models for the scalar displacement and the deviation. Once a target's potential movement is predicted, we may make sleep scheduling decisions based on these probabilistic models: take a high probability to awaken nodes on a direction along which the target is highly probable to move, and take a low one to awaken nodes that are not likely to detect the target.

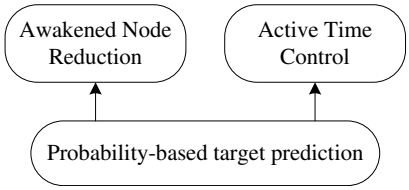


Fig. 1. PPSS design overview

Fig. 1 shows the three components of PPSS:

1) *Target prediction*. The proposed target prediction scheme consists of three steps: current state calculation, kinematics-based prediction and probability-based prediction. After calculating the current state, the kinematics-based prediction step calculates the expected displacement from the current location within the next sleep delay, and the probability-based prediction step establishes probabilistic models for the scalar displacement and the deviation.

2) *Awakened node reduction*. The number of awakened nodes is reduced with two efforts: controlling the scope of awake regions, and choose a subset of nodes in an awake region.

3) *Active time control*. Based on the probabilistic models that are established with target prediction, PPSS schedules an awakened node to be active, so that the probability that it detects the target is close to 1.

4 TARGET PREDICTION

In the real world, a target's movement is subject to uncertainty, while at the same time it follows certain rules of physics. This apparent contradiction is because: 1) at each instant or during a short time period, there is no significant change on the rules of a target's motion, therefore the target will approximately follow kinematics rules; 2) however, a target's long term behavior is uncertain and hard to predict, e.g., a harsh brake or a sharp turn cannot be predicted completely with kinematics rules. In fact, even for a short term, it is also difficult to accurately predict a target's motion purely with a physics-based model. However, the prediction is absolutely helpful for optimizing the energy efficiency and tracking performance tradeoff. Thus, we consider a probabilistic model to handle as many possibilities of change of the actual target motion as possible.

4.1 Overview

In this paper, we denote some of the vectors in a 2-dimensional plane with polar coordinates, for example, $\overrightarrow{X} = (X, \theta)$, where $X = \|\overrightarrow{X}\|$ is its polar radius and $\theta \in (-\pi, \pi]$ is the polar angle. In a real deployment, we may simply assign the four directions, south, east, north and west respectively as $-\frac{\pi}{2}$, 0, $\frac{\pi}{2}$ and π. As long as there is no ambiguity, we do not explicitly distinguish polar coordinates from Cartesian coordinates.

A target's movement status is a continuous function of time. However, the estimation for a target's movement status is a discrete time process. The surveillance system can only estimate the target states at some time points, and predict the future motion based on the estimation results. Thus, we assume that PPSS estimates the target states at time points $\{t_n | n \in N\}$ ($t_i < t_j \ for \ \forall \ i < j \in N$), and define the state vector to represent the target motion state:

Definition 1 (State Vector): For each time point t_n, the state vector is defined as $State(n) = (t_n, x_n, y_n, \overrightarrow{v_n}, \overrightarrow{a_n})$, where (x_n, y_n) is the target position, $\overrightarrow{v_n} = (v_n, \theta_n)$ and $\overrightarrow{a_n}$ are respectively the average velocity vector and the average acceleration vector of the target during (t_{n-1}, t_n), v_n is the scalar speed and θ_n is the moving direction.

Based on this definition, PPSS predicts the potential motion of a target at the time point t_n in three steps:

1) *Current state calculation*. Based on $State(n-1)$ and the current position (x_n, y_n) that is assumed to be obtained by sensing or by calculating, PPSS calculates the current speed v_n, direction θ_n and acceleration $\overrightarrow{a_n}$, and finally composes the current state vector $State(n)$.

2) *Kinematics-based prediction*. Based on kinematics rules, PPSS predicts $\overrightarrow{v_{n+1}}'$ and the displacement $\overrightarrow{S_{n+1}}'$ (i.e., the displacement vector during (t_n, t_{n+1})). We denote the predicted value of a variable as its name with a prime symbol, e.g., $\overrightarrow{S_{n+1}}'$ is the predicted $\overrightarrow{S_{n+1}}$.

3) *Probability-based prediction*. When the displacement $\overrightarrow{S_{n+1}}'$ is described with its polar coordinate $\overrightarrow{S_{n+1}}' = (S_{n+1}, \Delta_{n+1})$, we establish probabilistic models for the scalar displacement S_{n+1} (i.e., the polar radius) and the deviation Δ_{n+1} (i.e., the polar angle).

In the default duty cycling mode, the communication among nodes suffer a sleep delay with a MAC protocol like B-MAC, where a sending node

broadcasts the preamble no less than the length of a toggling period to guarantee that each duty cycling receiver can hear it [29]. We suppose the sleep delay exactly as TP for simplification, i.e., $t_{n+1} = t_n + TP$.

4.2 Current State Calculation

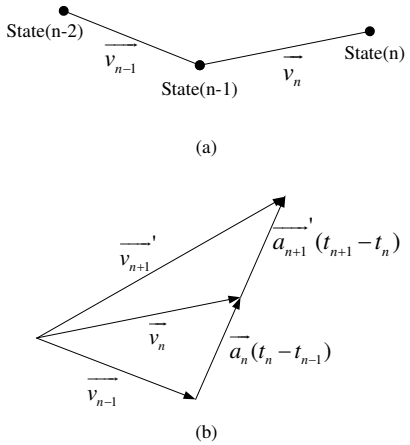


Fig. 2. State calculation and kinematics-based prediction

First, we calculate $State(n)$ based on $State(n-1)$ at the current time point t_n. Fig. 2(a) shows the target motion at three continuous time points, and Fig. 2(b) illustrates the change of the target's velocity and its acceleration. As assumed, the target's current location (x_n, y_n) can be determined by sensing or calculating with existing algorithms. Then, $\overrightarrow{v_n}$ and $\overrightarrow{a_n}$ can be computed as,

$$\begin{cases} v_n &= \frac{\sqrt{(y_n - y_{n-1})^2 + (x_n - x_{n-1})^2}}{t_n - t_{n-1}} \\ \theta_n &= \begin{cases} \arctan \frac{y_n - y_{n-1}}{x_n - x_{n-1}} & , \quad x_n \neq x_{n-1} \\ 0 & , \quad x_n = x_{n-1} \end{cases} \\ \overrightarrow{a_n} &= \frac{\overrightarrow{v_n} - \overrightarrow{v_{n-1}}}{t_n - t_{n-1}} \end{cases} \quad (1)$$

We didn't substitute $(t_n - t_{n-1})$ with TP, because the actual time point for sampling t_n depends on whether or not the target is physically detected. TP is only used for the prediction. Here we assume that sensor nodes are time synchronized locally in a small range, so that the received t_{n-1} may be used in the calculation. Local time synchronization may be easily achieved with a protocol such as RBS [32], or simply with HELLO message exchange [33].

Note that this calculation is based on the previous observation. If this is the first time that the target is detected and $State(n-1)$ does not exist, the calculation and the prediction will be delayed to the next time point.

4.3 Kinematics-based Prediction

Then, we utilize kinematics rules to predict $\overrightarrow{v_{n+1}}'$, $\overrightarrow{S_{n+1}}'$, and construct $State(n+1)'$. For simplifying the computation, we assume that the acceleration remains unchanged as $\overrightarrow{a_n}$ during (t_n, t_{n+1}). We argue this is a reasonable assumption, because the acceleration is the second derivative of displacement, and its change rate (i.e., the jerk) is the third derivative of displacement, which can usually be ignored in the displacements Taylor polynomial. Then

$$\begin{cases} \overrightarrow{a_{n+1}}' &= \overrightarrow{a_n} \\ \overrightarrow{v_{n+1}}' &= \overrightarrow{v_n} + \overrightarrow{a_{n+1}}' \cdot TP \\ \overrightarrow{S_{n+1}}' &= \overrightarrow{v_{n+1}}' \cdot TP + \frac{1}{2}\overrightarrow{a_{n+1}}' \cdot TP^2 \\ \overrightarrow{(x_{n+1}, y_{n+1})}' &= \overrightarrow{(x_n, y_n)} + \overrightarrow{S_{n+1}}' \end{cases} \quad (2)$$

Once time moves from t_n to t_{n+1}: 1) $\overrightarrow{(x_{n+1}, y_{n+1})}'$ will be replaced with $\overrightarrow{(x_{n+1}, y_{n+1})}$ measured at time point t_{n+1}; and 2) $\overrightarrow{a_{n+1}}'$ and $\overrightarrow{v_{n+1}}'$ will be replaced with the actual values $\overrightarrow{a_{n+1}}$ and $\overrightarrow{v_{n+1}}$ calculated in step 1 at time point t_{n+1}.

4.4 Probability-based Prediction

At the third step, we setup probabilistic models for random variables S_{n+1} and Δ_{n+1} of the predicted displacement $\overrightarrow{S_{n+1}}'$.

Suppose that S_{n+1} is Gaussian, i.e., $S_{n+1} \sim N(\mu_{S_{n+1}}, \sigma^2_{S_{n+1}})$. The mean is calculated as $\mu_{S_{n+1}} = \|\overrightarrow{S_{n+1}}'\| = \|\overrightarrow{v_{n+1}}' \cdot TP + \frac{1}{2}\overrightarrow{a_{n+1}}' \cdot TP^2\|$ when the acceleration remains unchanged. Next, we determine $\sigma_{S_{n+1}}$ based on the "68-95-99.7 rule" of Gaussian distribution [34]. During the sleep delay TP, the scalar speed is likely to change between $\|\overrightarrow{v_n}\|$ and $\|\overrightarrow{v_{n+1}}'\|$. Thus, S_{n+1} is likely to change between $S_A = \|\overrightarrow{v_n} \cdot TP\|$ and $S_B = \|\overrightarrow{v_{n+1}}' \cdot TP\|$, and $\mu_{S_{n+1}}$ is likely to fall in the interval (S_A, S_B) or (S_B, S_A) depending on the included angle between $\overrightarrow{v_{n+1}}'$ and $\overrightarrow{a_{n+1}}'$. Simply assigning the standard deviation of S_{n+1} with $\sigma_{S_{n+1}} = |\mu_{S_{n+1}} - S_A|$, the probability of $S_{n+1} \in (S_A, S_B)$ or $S_{n+1} \in (S_B, S_A)$ will be approximately 68%. Therefore, we setup the probabilistic model for the scalar displacement as $S_{n+1} \sim N(\mu_{S_{n+1}}, \sigma^2_{S_{n+1}})$ where

$$\begin{cases} \mu_{S_{n+1}} &= \|\overrightarrow{v_{n+1}}' \cdot TP + \frac{1}{2}\overrightarrow{a_{n+1}}' \cdot TP^2\| \\ \sigma^2_{S_{n+1}} &= (\|\overrightarrow{v_{n+1}}' \cdot TP + \frac{1}{2}\overrightarrow{a_{n+1}}' \cdot TP^2\| - \|\overrightarrow{v_{n+1}}' \cdot TP\|)^2 \end{cases} \quad (3)$$

Next, we establish a linear model for Δ_{n+1}. Assuming that the target holds the identical probability to turn left or right, we configure the probability density function of Δ_{n+1} as Equation 4, which is also shown in Fig. 3.

$$f_{\Delta_{n+1}}(\delta) = \begin{cases} -\frac{q}{p}\delta + q & , \quad (\delta \geq 0) \\ \frac{q}{p}\delta + q & , \quad (\delta < 0) \end{cases} \quad (4)$$

In the linear model, $p \leq \pi$ and q are coefficients, which can be determined with two conditions: the total probability (i.e., the area of the triangle) is equal to 1 (i.e., $pq = 1$); $E[\Delta_{n+1}] = 0$, thus $\sigma^2_{\Delta_{n+1}} =$

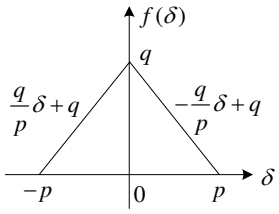


Fig. 3. Probability density distribution of Δ_{n+1}

$E[\Delta_{n+1}^2] = \int_{-p}^{p} \delta^2 f(\delta) d\delta$. Then, the coefficients p and q can be determined with a given $\sigma_{\Delta_{n+1}}$:

$$\begin{cases} p &= \sqrt{6}\sigma_{\Delta_{n+1}} \\ q &= \frac{\sqrt{6}}{6\sigma_{\Delta_{n+1}}} \end{cases}$$

The variance $\sigma^2_{\Delta_{n+1}}$ can be configured by the application or dynamically computed regarding to the acceleration. For example, by assigning $\sigma_{\Delta_{n+1}}$ with the calculated maximum deviation angle δ_{max}, the probability of $\Delta_{n+1} \in (-\delta_{max}, \delta_{max})$ will be approximately 68% (based on the "68-95-99.7 rule" of Gaussian distribution [34]).

In fact, it is also possible to model the deviation Δ_{n+1} with other probabilistic distributions, e.g., Gaussian distribution. However, the difference between the influence of two distributions on the performance of PPSS is very slight [35]. On the contrary, Gaussian distribution may introduce a significantly higher computational complexity than the linear distribution. Thus, we do not take this option.

5 Energy Conservation

In this section, we first introduce the concept of awake region and the proactive wake-up process with awake regions, then describe the approaches for reducing the energy consumption.

5.1 Proactive Wake-up with Awake Regions

An awake region is defined as the region that a target may traverse in a next short term, which should be covered probabilistically by active nodes. The term awake region, as we use it, is similar to the concept of a cluster used in the network architecture work (e.g., [10], [36]) in that it encompasses some of a cluster's functions. However, unlike a cluster's head, neither an alarm node aggregates data from member nodes of the awake region, nor it imposes any control over members. An alarm node's responsibility here is just to broadcast an alarm message on detecting a target. In fact, an awake region is only a virtual concept. No functions are built upon this concept, except for the selection of awakened nodes.

Unlike cluster management, the life cycle of awake regions is described as follows:

1) *Creation*. On detecting a target, a sensor node will check its own status to determine if it is an awakened node in an existing awake region. If yes, it justifies if the target is leaving the current awake region. If no previous awake region exists or if the target is leaving the current awake region, the node runs an alarm node election algorithm, e.g. [37]. If this node is elected as the alarm node, it broadcasts an alarm message to all the candidate nodes. On receiving this alarm message, each candidate node individually decides if it is in the scope of this awake region and whether or not to schedule the sleep pattern. Finally, a new awake region is formed when every awakened node schedules their sleep patterns specifically for the approaching target.

2) *Maintenance*. If an awake region exists and the target is not going to move out of the current awake region, the node keeps active without sleep scheduling operations, and the awake region remains unchanged.

3) *Dismissal*. As time progresses, the sleep patterns of awakened nodes will automatically recover back to the default pattern, thus the awake region will be dismissed automatically. There is no explicit dismissal mechanism needed.

This is a distributed process: based on the alarm broadcasting, each node makes the sleep scheduling decision, and returns to the default duty cycling mode all by itself. An alarm message that is used to make this decision contains the following information:

- ID and the position of the alarm node (id_r, x_r, y_r);
- The state vector $State(n)$; and
- The prediction results, including $\overrightarrow{S_{n+1}}'$, $\mu_{S_{n+1}}$, $\sigma_{S_{n+1}}$, and $\sigma_{\Delta_{n+1}}$.

The approach for electing an alarm node of [37] is as follows. Upon detection, each node broadcasts a DETECTION message to nodes nearby containing a time stamp recording when the detection is declared. Then it checks all the DETECTION messages received from nodes nearby within an interval, and compares the time stamps of other nodes with its own. Nodes that detect a neighbor node's time stamp is earlier than its own simply keep silent. In fact, a simpler approach also works well. Without any alarm election algorithm used, multiple alarm messages may be broadcast from multiple nodes that detect the same target. Then on receiving the first alarm message, a neighbor node may stop its own alarm broadcasting, and simply ignore the following ones sent by nodes that are within a $2r$ distance from the first alarm node, as these alarms may be considered as for the same target.

5.2 Awakened Node Reduction

Usually, a sensor node's transmission range R is far longer than its sensing range r. Thus when the nodes are densely deployed to guarantee the sensing coverage, a broadcast alarm message will reach all the neighbors within the transmission range. However,

some of these neighbors can only detect the target with a relatively low probability, and some others may even never detect the target. Then the energy consumed for being active on these nodes will be wasted. A more effective approach is to determine a subset among all the neighbor nodes to reduce the number of awakened nodes.

During the sleep delay, the target may move away from the alarm node for a distance. Then it is unnecessary for nodes within this distance to wake up, since the target has already passed by. Meanwhile, all the nodes in an awake region must in the one-hop transmission range of the alarm node. Therefore, an awake region should be in a ring shape, i.e., the part between two concentric circles.

Beyond the effort that limits the awakened nodes within an awake region, the number of awakened nodes can be further reduced by choosing only some nodes in the awake region as awakened nodes. Based on our prediction on the target's moving directions, the probabilities that the target moves along various directions are different. Obviously the number of awakened nodes along a direction with a lower probability could be less than the number along a direction with a higher probability. By choosing an awakened node based on a probability related to the moving directions, awakened nodes can be reduced significantly.

5.2.1 Constrain the Awake Region Scope

Letting d denote the distance of an awakened node from the alarm node, we next determine an awake region's scope by deciding the value scope of d. Here, we make the same approximate assumption that the target's position is exactly the alarm node's position to simplify the computation.

As previously discussed, the target may move by S_{n+1} during the sleep delay TP. If we set $d \geq \mu_{S_{n+1}} - \sigma_{S_{n+1}}$, the probability that awakened nodes cannot cover the target after a sleep delay will be less than $\frac{(1-68\%)}{2} = 16\%$ (according to the "68-95-99.7 rule" of Gaussian distribution [34]). Moreover, it is obvious that $d \leq R$ because nodes outside of the alarm node's transmission range cannot be awakened. Therefore we determine the scope of an awake region as $\max\{\mu_{S_{n+1}} - \sigma_{S_{n+1}}, 0\} \leq d \leq R$. Thus the number of nodes in an awake region is $\rho\pi\left[R^2 - \max\{\mu_{S_{n+1}} - \sigma_{S_{n+1}}, 0\}^2\right]$, where ρ is the node density.

5.2.2 Awakened Nodes Selection

So far the computation of an awake region's scope depends on the target's scalar speed only. Moreover, the decrement percentage of the number of awakened nodes is only $\frac{\max\{\mu_{S_{n+1}} - \sigma_{S_{n+1}}, 0\}^2}{R^2}$ (e.g., 6.25% when $R = 60$, $\mu_{S_{n+1}} = 20$, and $\sigma_{S_{n+1}} = 5$), which is not significant enough for enhancing energy efficiency.

As discussed previously, only some of the member nodes in an awake region need to be awakened. By taking into account the prediction results on moving directions, we can further reduce the number of awakened nodes in an awake region so as to save more energy than solely constraining the scope of an awake region.

Since the probability that a target moves along the direction of $\overrightarrow{S_{n+1}}'$ (i.e. $E[\Delta_{n+1}]$), denoted as θ, is the highest, we force all the nodes on this direction to be awakened. As Δ_{n+1} decreases on other directions, the number of awakened nodes on those directions can also be decreased. We define the probability that a candidate node on the direction $(\theta + \delta)$ reschedules its sleep pattern (i.e., becomes an awakened node) as

$$P_{ss}(\delta) = \frac{f_{\Delta_{n+1}}(\delta)}{f_{\Delta_{n+1}}(0)} = \begin{cases} -\frac{1}{p}\delta + 1 & , \quad (\delta \geq 0) \\ \frac{1}{p}\delta + 1 & , \quad (\delta < 0) \end{cases}$$

where "ss" means sleep scheduling.

Then, the total number of awakened nodes in an awake region would be

$$\begin{aligned} N &= \int_{-\pi}^{\pi} P_{ss}(\delta) \cdot \frac{\rho\pi(R^2 - \max\{\mu_{S_{n+1}} - \sigma_{S_{n+1}}, 0\}^2)}{2\pi} d\delta \\ &= \rho(R^2 - \max\{\mu_{S_{n+1}} - \sigma_{S_{n+1}}, 0\}^2) \cdot \int_0^{\pi} \big(-\frac{1}{p}\delta + 1\big) d\delta \\ &= \frac{\sqrt{6}}{2}\rho\sigma_{\Delta_{n+1}}(R^2 - \max\{\mu_{S_{n+1}} - \sigma_{S_{n+1}}, 0\}^2) \end{aligned}$$

As an example, the number of awakened nodes of PPSS is only approximately 19% of that of the Circle scheme when $R = 60$, $\mu_{S_{n+1}} = 20$, $\sigma_{S_{n+1}} = 5$, and $\sigma_{\Delta_{n+1}} = \frac{\pi}{6}$. In another word, the energy consumption of PPSS is only about 19% of that of the Circle scheme.

5.3 Active Time Control

After reducing the number of awakened nodes, energy efficiency can be enhanced further by scheduling the sleep patterns of awakened nodes, as not all the awakened nodes need to keep active all the time. We schedule the sleep patterns of awakened nodes by setting a start time and an end time of the active period. Out of this active period, awakened nodes do not have to keep active. Therefore, the time that an awakened node has to keep active could be reduced compared with the Circle scheme.

At the moment that an awakened node receives the alarm message (i.e. after the sleep delay, we denote this time point as $t_{alarmed}$), the relationship between the awakened node's position and the distribution of the target's displacement length during a sleep delay is shown in Fig. 4. In the figure, 0 means the position of the alarm node. According to the relative positions of the awakened node and the target's expected position after the sleep delay, we make the sleep scheduling decisions as follows.

When $\mu_{S_{n+1}} \geq d - r$, the awakened node is required to wake up immediately (i.e. at $t_{alarmed}$) since it is

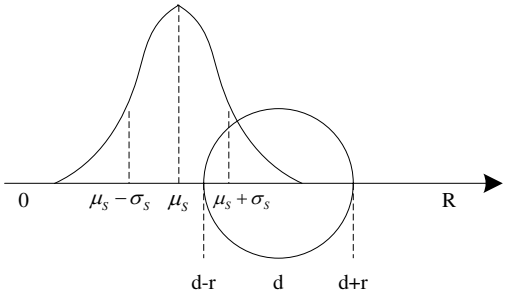


Fig. 4. Relationship between S_{n+1} and d

expected that the target has probably entered its sensing range. When $\mu_{S_{n+1}} < d-r$, the awakened node is required to wake up at $t_{alarmed}+\frac{d-r-\mu_{S_{n+1}}}{TS}$, where we suppose $TS = \frac{\mu_{S_{n+1}}}{TP}$ to be the average speed in the awake region. In both cases, the awakened node needs to keep active until $t_{alarmed} + \frac{d+r-(\mu_{S_{n+1}}-\sigma_{S_{n+1}})}{TS}$. At this time point, the probability that the target has moved out of the sensing range of the awakened node will be greater than $1 - \frac{(1-68\%)}{2} = 84\%$ (according to the "68-95-99.7 rule" of Gaussian distribution [34]). For the convenience of discussion, we denote $t_{start} = t_{alarmed} + \frac{max\{d-r-\mu_{S_{n+1}},0\}}{TS}$ and $t_{end} = t_{alarmed} + \frac{d+r-(\mu_{S_{n+1}}-\sigma_{S_{n+1}})}{TS}$. In summary, the rescheduled active period of an awakened node is $\left[t_{start}, t_{end}\right]$, where

$$\begin{cases} t_{start} &= t_{alarmed} + \frac{max\{d-r-\mu_{S_{n+1}},0\}}{TS} \\ t_{end} &= t_{alarmed} + \frac{d+r-(\mu_{S_{n+1}}-\sigma_{S_{n+1}})}{TS} \end{cases} \quad (5)$$

And the time of keeping active is,

$$T_{active} = \frac{min\{2r+\sigma_{S_{n+1}}, d+r-(\mu_{S_{n+1}}-\sigma_{S_{n+1}})\}}{TS} \quad (6)$$

Once a node is sleep-scheduled, it will keep active until it returns to the default duty cycling mode after the scheduled active time. Thus, other than the timer for default duty cycling (or *default timer*), a new wake-up timer (or *tracking timer*) is needed to end the scheduled state.

6 Distributed Algorithm

For the actual implementation, all of these mechanisms presented in Section 5 have to be distributed on each sensor node. In this section, we present detailed algorithm descriptions for PPSS protocol in three procedures.

Procedure 1 is a handler for the event of detecting a target, which can be triggered by an interrupt that is raised on sensing something.

For the formation frequency of awake regions, the MCTA algorithm [12] uses a "refresh time" concept. Instead, we use the target's motion trend as the criterion: when the target moves close to the edge of the current awake region, a sensor node, which detects the target is leaving and is elected as the alarm node, broadcasts an alarm message to wake up neighbors and form a new awake region.

Procedure 1 *OnDetectingTarget()* — Triggered when detecting a target

1: **if** (I am scheduled to be active) **then**
2: **if** (The target is *NOT* leaving the current awake region) **then**
3: return;
4: **end if**
5: **end if**
6: (Optional:) Run an alarm election algorithm;
7: **if** (I am selected as the alarm node) **then**
8: Calculate $\overrightarrow{v_n}$ and $\overrightarrow{a_n}$ with Equation 1;
9: Predict $\overrightarrow{S_{n+1}}'$ with Equation 2;
10: Compute $\mu_{S_{n+1}}, \sigma_{S_{n+1}}, \sigma_{\Delta_{n+1}}$ with Equation 3;
11: Broadcast $id_r, x_r, y_r, State(n), \overrightarrow{S_{n+1}}', \mu_{S_{n+1}}, \sigma_{S_{n+1}}, \sigma_{\Delta_{n+1}}$;
12: **end if**
13: return;

Procedure 2 describes a sensor node's actions upon receiving an alarm message. This procedure can also be implemented as an interrupt handler.

Procedure 2 *OnAlarmMsg()*—Triggered when receiving an alarm message

1: Compute the distance d to the alarm node;
2: **if** ($d < \mu_{S_{n+1}} - \sigma_{S_{n+1}}$) **then**
3: return;
4: **end if**
5: Compute δ with the alarm node position, my position and $\overrightarrow{S_{n+1}}'$;
6: Generate a random number $random = [0, 1]$;
7: **if** ($random > P_{ss}(\delta)$) **then**
8: return;
9: **end if**
10: Compute t_{start} and t_{end} with Equation 5;
11: SetTrackingTimer(t_{start});
12: return;

In step 2 of Procedure 2, the node determines whether or not it is in the scope of the awake region. In step 7, the node decides whether to be an awakened node or not. Finally in step 11, the tracking timer is set so that the node can wake up at the scheduled time point.

Procedure 3 describes the tracking timer processing procedure, which controls the scheduled wake-up/sleep and the mode switch.

The computation workload of PPSS protocol is mainly located in steps 8, 9, 10 of Procedure 1, i.e., the calculation for $\overrightarrow{v_n}$, $\overrightarrow{a_n}$, and the prediction for $\overrightarrow{S_{n+1}}', \mu_{S_{n+1}}, \sigma_{S_{n+1}}, \sigma_{\Delta_{n+1}}$. Thus the computation workload is aggregated on the alarm node, which is

Procedure 3 *OnTrackingTimer()* — Triggered when the tracking timer is out

```
1: if (mode == "default") then
2:     mode = "tracking";
3:     SuspendDefaultTimer();
4:     SetTrackingTimer(t_end);
5: else
6:     mode = "default";
7:     ResumeDefaultTimer();
8: end if
9: return;
```

more energy efficient than a distributed computation.

7 Experimental Evaluation

We evaluated PPSS protocol with both simulation-based and implementation-based experiments: the simulation was conducted in an environment developed in C++, and the prototype implementation was developed based on TelosB motes [15] and TinyOS version 2.1.1 [16].

PPSS was compared to Circle scheme [6] and MCTA algorithm [12]. The primary difference among the three protocols is how they reduce the energy consumption for proactive wake-up: 1) Circle awakens all the one hop neighbors of the alarm node, thus consumes the most energy; 2) MCTA compresses the area where nodes are awakened, but still awakens every node in the area; and 3) PPSS compresses the awake region, awakens selected nodes only, and further reduces their active time.

7.1 Performance Metrics

Before reporting our evaluation results, we first define the metrics used to estimate energy efficiency and tracking performance. In the experiments, all of these metrics were examined for each target intrusion case.

1) *Energy efficiency*. Since the energy consumption of sleep scheduling is highly relative to the position where the target is detected, the energy consumptions on different nodes may vary significantly. Given this inequity, the network lifetime—time until the first sensor node runs out of power—will be a less useful metric. Instead, we use the network-wide *extra energy* (EE) as the criterion of energy efficiency for sleep scheduling, which is defined as:

$$EE = \sum_i EE_i = \sum_i (E_{scheduled} - E_{default})$$

where $E_{default}$ is the energy consumption of node i for idle listening when no target is detected, and $E_{scheduled}$ is the total energy of node i consumed for sleep scheduling when a target is detected (e.g., for proactive wake-up, prolonged active time etc.). Both of them are measured during the same tracking period. As we focus on keeping tracking a target instead of data collection, neither of them involves the communication energy for propagating target information towards sink nodes.

2) *Tracking performance (for the simulation)*. The tracking delay is one of the most important performance metrics for tracking. Since tracking is a process of continuous detections, we describe the tracking delay with *average detection delay* (AD), which is defined as the trajectory-wide average of escape times:

$$AD = E[\Delta T]$$

where ΔT is the interval between the time when the target enters the surveillance field or gets lost, and the time when it is detected for the next time. Before the target is detected for the first time, PPSS protocol is not started and all the nodes work in the default duty cycling mode. Thus, the initial detection delay [4] is out of the scope of PPSS's performance. We measure AD only after the first detection.

3) *Tracking performance (for the implementation)*. In the prototype implementation, it is usually difficult to achieve precise time synchronization. Although we need to implement time synchronization to run PPSS, the detection delay measurement is probably not precise enough to evaluate the tracking performance. Instead, we use *escape distance percentage* (EDP) as an alternative of AD, which is defined as the percentage of a target's escape distance in the total trajectory:

$$EDP = \frac{\sum D_{escape}}{D_{total}}$$

where D_{escape} is the distance that none of the nodes can detect the target, and D_{total} is the total length of the target's trajectory. Thus, a low escape distance percentage will mean a good coverage on the target's trajectory.

7.2 Simulation-based Evaluation

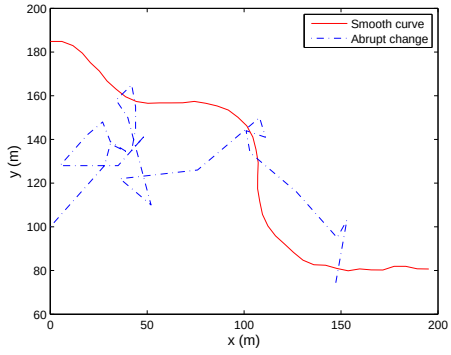


Fig. 5. Target route examples

In the simulation, we evaluated the influence of the following factors on EE and AD of three protocols: node density ρ, target speed v, the $\frac{R}{r}$ ratio (i.e., the

ratio of the communication radius to the sensing radius), localization error ϵ, and target movement model. We randomly deployed up to $1,600$ nodes in a $200m \times 200m$ area to track a target that moves at a speed no greater than $30\ m/s$. R/r varies from 2 to 6, and ϵ varies from $1\ m$ to $5\ m$. In addition, the target may move in a smooth curvilinear trajectory or with abrupt direction changes, for which two example moving routes are shown in Fig. 5.

When we study the performance against a certain factor, the other factors remain as the default values, which are respectively $\rho = 1.5\ node/100m^2$, $v = 18\ m/s$, $R/r = 6$, $\epsilon = 3\ m$, and the smooth curvilinear movement. For each configuration case, we repeated the experiment for 100 times and recorded the average as the final result.

TABLE 1
Energy consumption rates

Status	Energy consumption rate (unit)
Active (P_{active})	$9.6\ (mJ/s)$
Transmit (P_{send})	$720\ (nJ/bit), 5.76\ (mJ/Byte)$
Receive (P_{rcv})	$110\ (nJ/bit), 0.88\ (mJ/Byte)$
Sleep (P_{sleep})	$0.33\ (mJ/s)$

Besides these factors, we configure the other parameters including $TP = 1\ s$, $DC = 10\%$, and the energy consumption rates of Mica2 platform [38], [39] shown in Table 1. In Table 1, we did not list the energy consumption rate for instruction execution, as it is difficult to measure in the simulation. Given that PPSS may introduce a higher computational complexity than Circle and MCTA, we increased the energy consumption rate for PPSS's active state by 20%, in order to achieve a fair comparison.

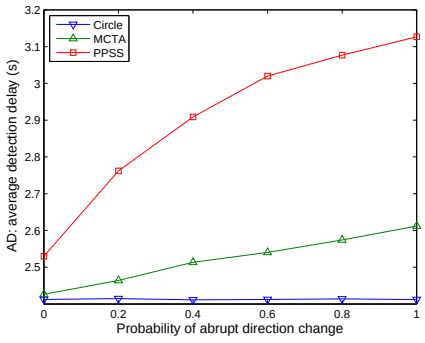


Fig. 6. AD at various probabilities of abrupt direction change

The evaluation results under various node densities, target speeds, $\frac{R}{r}$ ratios, and localization errors are shown in Fig. 7. Next, we discuss the detailed simulation results:

1) ***Node density***. Fig. 7(a) and Fig. 7(b) show the comparison of three protocols at various node densities. As the node density increases, EE increases, and AD decreases, because an increasing number of nodes are proactively awakened to track the target.

2) ***Target speed***. Fig. 7(c) and Fig. 7(d) show the evaluation results at various target speeds. We observe that as the target moves faster, both EE and AD decrease. The reason that EE decreases is that a faster target will stay shorter in the alarm node's communication range, thereby require awakened nodes to keep active for a shorter time, than a slower target. AD will also decrease, because when the target moves faster, it will traverse the sensing range of more sensor nodes in the same period of time, which in turn increments the probability of being detected.

3) $\frac{R}{r}$ ***ratio***. In the simulation, we fixed R and decreased r to increase the $\frac{R}{r}$ ratio. The performance of three protocols at various ratio values is shown in Fig. 7(e) and Fig. 7(f). When r decreases, the active time of awakened nodes will decrease according to Equation 6. Thus, the energy consumption will decrease, too. At the same time, a decreasing r will decrement the probability of detecting a target, therefore increment the detection delay.

4) ***Localization error***. Fig. 7(g) and Fig. 7(h) show the evaluation results at various localization errors. We observe that the impact of the localization error on Circle scheme and MCTA is slight, because they awaken all the nodes in an area or a contour. On the contrary, the detection delay of PPSS will increase as the error increases, because the target may deviate from awakened nodes just like in a default duty cycling network.

5) ***Abrupt direction change***. We also compared the performance of three protocols when the target moves with abrupt direction changes (for which a large δ_{max} in Section 4.4 is just an example). As their energy consumptions showed very similar curves as Fig. 7(g), we only show the average detection delays in Fig. 6. Since Circle scheme has no inclination on specific direction, its AD does not change significantly with the probability of abrupt direction change. On the contrary, both MCTA and PPSS will be influenced by the abrupt direction change of the target. Especially, the detection delay of PPSS will increase quickly to the level of the default duty cycling mode. Therefore, PPSS has limitation on slow targets with a high level of abrupt direction change.

Compared to MCTA, we observe that PPSS introduces an improvement of $25\% \sim 45\%$ on energy efficiency, at the expense of only $5\% \sim 15\%$ increase on detection delay. Among the three protocols, Circle scheme serves as the upper bound for both the energy consumption and the tracking performance. This is because: 1) Circle scheme awakens all the neighbor nodes within the alarm node's communication radius, thus consumes the most energy; and 2) it keeps all the awakened nodes active for a relatively long time, thus guarantees the best tracking performance.

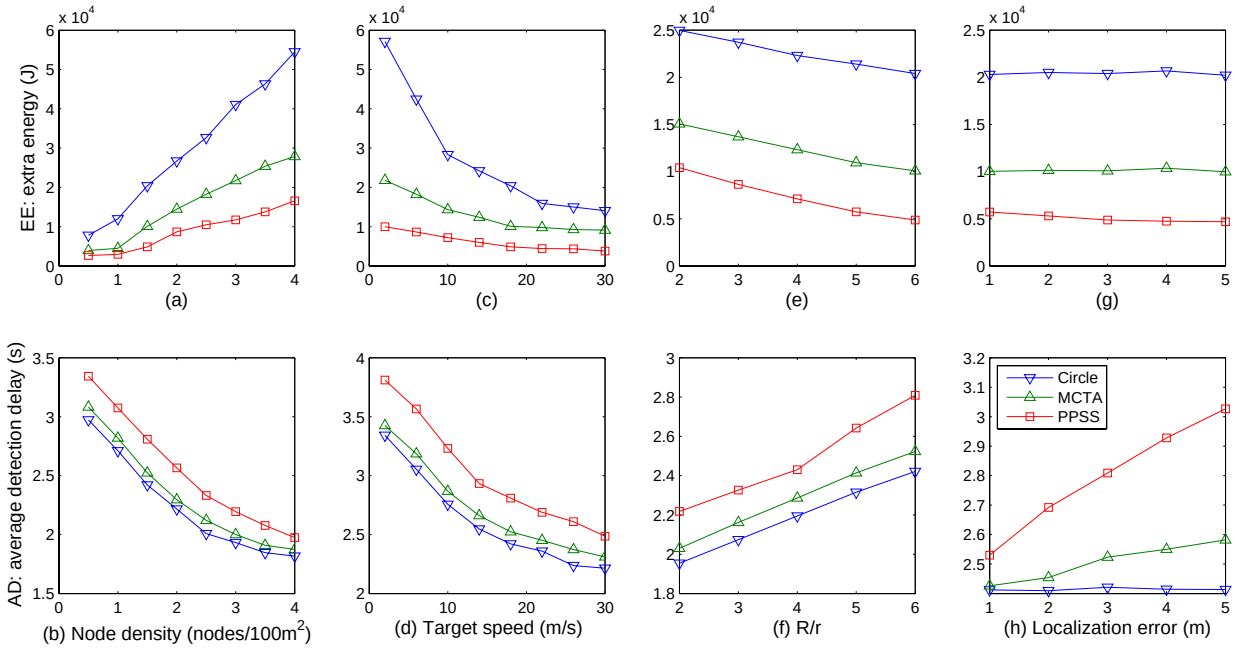


Fig. 7. Performance under various node densities, target speeds, $\frac{R}{r}$ ratios, and localization errors

7.3 Implementation-based Evaluation

Besides verifying the efficiency of PPSS, the implementation also exposes the design to real network environments with noises, and irregular sensing/communication radii.

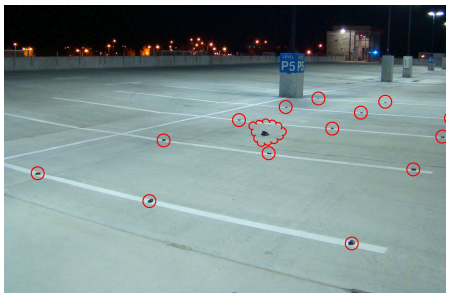


Fig. 8. Mote deployment

In an outdoor parking lot, we deployed 15 TelosB motes [15] in a 3×5 grid, each of which runs the competing protocols that we implemented on TinyOS version 2.1.1 [16]. As shown in Fig. 8, a toy car was remotely controlled to move in a line across the field.

Unlike the simulation, the real implementation requires actual solutions to many detailed issues[1]:

1) *Energy measurement*. Three of the commonly used energy measurement methods — battery emulation/simulation [41], on-board power meter [42], and online multimeter [43] — all depend on special hardware devices, and are subject to constraints [40]. Given this situation, we took the emulation approach instead: we record the number of operations, and the communicated data amount or the operating time of each operation, then calculate the energy consumption based on these raw data and the configured energy consumption rates [15].

2) *Target detection*. It is difficult for TelosB motes to actually detect a physical target, given its constrained sensing devices (i.e., light, temperature and humidity). Alternatively we emulate a mobile target with a specially designed mote, called *target node*, which broadcasts its positions (configured in a header file *a priori*) that other motes may "sense". The target node broadcasts short messages periodically at every 200 ms. Obviously, these extra messages will influence the application's regular communications, as they consume some wireless bandwidth. However, this is inevitable without the actual sensing devices. We tried to minimize this influence by shortening the message length to the least, i.e., the target's position only. During the experiments, the target node was attached to the remotely controlled toy car.

3) *Data collection*. The influence of experimental operations on regular operations of protocols should be minimized, because otherwise the experimental results will be less reliable. For this purpose, we stored experimental results temporarily on individual motes, and designed a data collection node to collect them after the experiments.

4) *Time synchronization*. Local time synchronization was implemented via HELLO message exchange [33].

Protocols under evaluation were implemented in the structure shown in Fig. 9. The target sensor component signals an event of detection when receiving

1. Limited by the paper space, we cannot specify every detail of the implementation. More information can be found in [40].

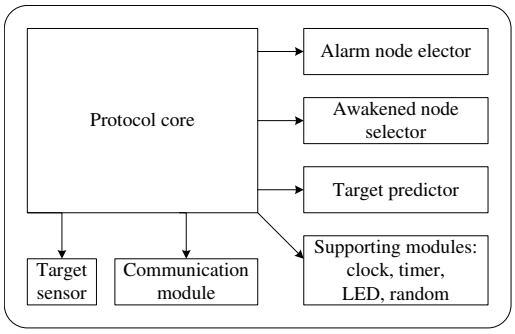


Fig. 9. Protocol structure

the target node's broadcast. The communication module communicates with neighbor motes. Once a target is detected, its potential movement is first predicted by the target predictor module. Then, the alarm node elector module elects a mote to broadcast the alarm message. When such an alarm message is received, the awakened node selector module decides if this mote should be awakened proactively. Besides these modules, we designed a few supporting modules, including local clock, timer, LED control, and random number generator. The protocol core module handles the protocol details. Each time a service of a functional module is needed, the protocol core module calls a command provided by that module, and possibly handles the events triggered by the command call. In this structure, three competing protocols are only distinguished from each other in the awakened node selector module and the core protocol module.

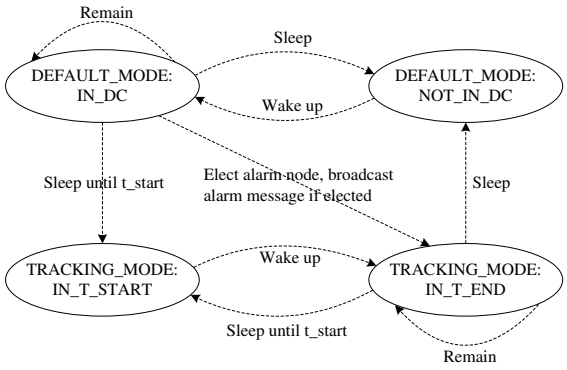


Fig. 10. PPSS protocol finite state machine

In the protocol core module of PPSS, we designed an internal finite state machine consisting of four states as shown in Fig. 10[2]:

1) Default mode/IN_DC: active in the default duty cycling mode;

2) Default mode/NOT_IN_DC: sleeping in the default duty cycling mode;

3) Tracking mode/IN_T_START: the sleep pattern is scheduled to sleep until t_{start}; and

2. Fig. 10 was simplified due to the length limit. Please refer to [40] for the complete version.

4) Tracking mode/IN_T_END: the sleep pattern is scheduled to keep active until t_{end}.

For each protocol under testing, the experiment was repeated five times. Unlike the simulation, we did the implementation-based experiment under a single deployment only. This is because that compared to the simulation, changing network configuration (e.g. node density) is more difficult in the implementation. For example, the transmission power level of motes' RF radio can only be configured as a series of discrete integer values, i.e., the communication radius cannot be configured to any number. Therefore, simply redeploying nodes in a different density, even in the same topology, may introduce a different connection status of nodes. Comparing the experimental results obtained from these different connection status will be less helpful.

As previously discussed, we evaluate EDP for tracking performance in the implementation, instead of AD used in the simulation.

TABLE 2
Implementation-based experiment results

Protocol		Circle	MCTA	PPSS
EE_{alarm} (mJ)		8.01	7.92	8.13
EE_{active} (mJ)		74.1	38.6	33
Total EE	μ (mJ)	82.11	49.52	41.13
	μ_{norm}	1.658	1	0.8306
	σ^2	13.31	11.20	7.110
	CI	(78.63–85.59)	(45.57–53.47)	(38.59–43.67)
EDP	μ (%)	14.0	19.6	20.4
	μ_{norm}	0.7143	1	1.041
	σ^2	2.15	2.83	2.82
	CI	(12.65–15.35)	(18.00–21.20)	(18.80–21.99)

In Table 2, we show sample means μ, normalized means μ_{norm} (to MCTA's results), variances σ^2, and 90% confidence intervals CI (assuming that the samples are normally distributed) of EE and EDP from five repeated experiments. EE_{alarm} is the energy for alarm messages, and EE_{active} represents the energy for scheduled wake-up. From the table, we observe:

1) Compared to Circle scheme, PPSS improves energy efficiency by 49.9%, with the cost of a 45.7% increase on EDP.

2) Compared to MCTA algorithm, PPSS improves energy efficiency by 16.9%, with the cost of a 4.1% increase on EDP.

3) In the extra energy consumed for sleep scheduling, the energy for alarm message communication takes a small part, and most of the energy is consumed for keeping motes active to wait for the approaching target. Just because PPSS reduces the number of awakened motes and their active time, energy efficiency is improved significantly.

In Fig. 11, we plot the extra energy consumption of individual motes to show the energy consumption distribution. The energy data was obtained from one sample of our experiments. Note that the extra energy

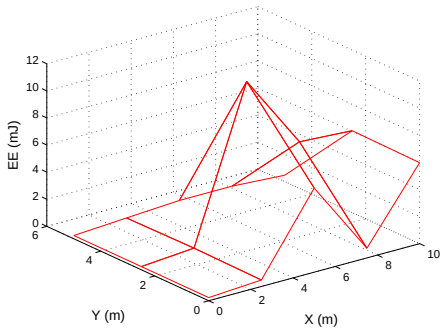


Fig. 11. Extra energy of motes with PPSS

includes the energy consumed for sleep scheduling only. Since the sink node neither sends alarm messages nor was sleep-scheduled, its extra energy was found zero.

8 Conclusion

In a duty-cycled sensor network, proactive wake-up and sleep scheduling can create a local active environment to provide guarantee for the tracking performance. By effectively limiting the scope of this local active environment (i.e., reducing low value-added nodes that have a low probability of detecting the target), PPSS improves the energy efficiency with an acceptable loss on the tracking performance. In addition, the design of PPSS protocol shows that it is possible to precisely sleep-schedule nodes without involving much physics.

Though the emulation is sometimes unavoidable, our prototype implementation can still provide more real and convincing results than the simulation. For example, besides exposing motes to real environmental noises and unstable links, the implementation itself can verify the rationality of the solutions, and the feasibility of applying them into the constrained resources of actual mote hardware platforms.

Except for the strengths, PPSS has limitations as well. First, it does not use optimization methods, i.e., PPSS imposes no performance constraints when reducing the energy consumption. Without performance constraints, it is difficult to configure the protocol toward the best energy-performance trade-off for a specific network environment. However, the optimization is difficult for PPSS, because it will involve many physics problems, which are out of this paper's scope. Instead, we make an experiment-based effort that evaluates the performance of PPSS under various conditions. Secondly, the prediction method of PPSS cannot cover special cases such as the target movement with abrupt direction changes. This is the expense that PPSS pays for the energy efficiency enhancement. Given these limitations, the potential future work includes optimization-based sleep scheduling and target prediction for abrupt direction changes.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks (Amsterdam, Netherlands: 1999)*, vol. 38, no. 4, pp. 393–422, 2002.

[2] Q. Cao, T. Yan, J. Stankovic, and T. Abdelzaher, "Analysis of target detection performance for wireless sensor networks," in *Intl Conference on Distributed Computing in Sensor Systems (DCOSS)*, 2005, pp. 276–292.

[3] G. Wittenburg, N. Dziengel, C. Wartenburger, and J. Schiller, "A system for distributed event detection in wireless sensor networks," in *IPSN '10: Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*. New York, NY, USA: ACM, 2010, pp. 94–104.

[4] T. He, P. Vicaire, T. Yan, L. Luo, L. Gu, G. Zhou, R. Stoleru, Q. Cao, J. A. Stankovic, and T. Abdelzaher, "Achieving real-time target tracking using wireless sensor networks," in *RTAS '06: Proceedings of the 12th IEEE Real-Time and Embedded Technology and Applications Symposium*, 2006, pp. 37–48.

[5] Q. Cao, T. Abdelzaher, T. He, and J. Stankovic, "Towards optimal sleep scheduling in sensor networks for rare event detection," in *Proceedings of the 4th international symposium on Information processing in sensor networks*, 2005, p. 4.

[6] C. Gui and P. Mohapatra, "Power conservation and quality of surveillance in target tracking sensor networks," in *Proceedings of the 10th annual international conference on Mobile computing and networking*, 2004, pp. 129–143.

[7] G. Lu, N. Sadagopan, B. Krishnamachari, and A. Goel, "Delay efficient sleep scheduling in wireless sensor networks," in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, vol. 4, March 2005, pp. 2470–2481.

[8] Y. Gu and T. He, "Data forwarding in extremely low duty-cycle sensor networks with unreliable communication links," in *SenSys '07: Proceedings of the 5th international conference on Embedded networked sensor systems*, 2007, pp. 321–334.

[9] Y. Wu, S. Fahmy, and N. Shroff, "Energy efficient sleep/wake scheduling for multi-hop sensor networks: Non-convexity and approximation algorithm," in *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, May 2007, pp. 1568–1576.

[10] X. Wang, J.-J. Ma, S. Wang, and D.-W. Bi, "Cluster-based dynamic energy management for collaborative target tracking in wireless sensor networks," *Sensors*, vol. 7, pp. 1193–1215, 2007.

[11] J. Fuemmeler and V. Veeravalli, "Smart sleeping policies for energy efficient tracking in sensor networks," *Signal Processing, IEEE Transactions on*, vol. 56, no. 5, pp. 2091–2101, May 2008.

[12] J. Jeong, T. Hwang, T. He, and D. Du, "Mcta: Target tracking algorithm based on minimal contour in wireless sensor networks," in *INFOCOM*, 2007, pp. 2371–2375.

[13] X. Wang, J.-J. Ma, S. Wang, and D.-W. Bi, "Prediction-based dynamic energy management in wireless sensor networks," *Sensors*, vol. 7, no. 3, pp. 251–266, 2007.

[14] Y. Xu, J. Winter, and W.-C. Lee, "Prediction-based strategies for energy saving in object tracking sensor networks," in *Mobile Data Management, 2004. Proceedings. 2004 IEEE International Conference on*, 2004, pp. 346–357.

[15] CrossBow, "Telosb data sheet," http://www.willow.co.uk/TelosB_Datasheet.pdf.

[16] "Tinyos." [Online]. Available: http://www.tinyos.net/

[17] Y. Zhuang, J. Pan, and L. Cai, "Minimizing energy consumption with probabilistic distance models in wireless sensor networks," March 2010, pp. 1–9.

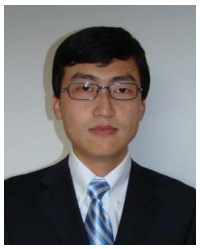
[18] T. He, P. Vicaire, T. Yan, Q. Cao, and G. Z. et al., "Achieving long-term surveillance in vigilnet," *INFOCOM*, 2006.

[19] C. Sengul, M. J. Miller, and I. Gupta, "Adaptive probability-based broadcast forwarding in energy-saving sensor networks," *ACM Trans. Sen. Netw.*, vol. 4, pp. 6:1–6:32, April 2008.

[20] Y. M. Lu and V. W. S. Wong, "An energy-efficient multipath routing protocol for wireless sensor networks: Research articles," *Int. J. Commun. Syst.*, vol. 20, no. 7, pp. 747–766, 2007.

[21] J. Aslam, Z. Butler, F. Constantin, V. Crespi, G. Cybenko, and D. Rus, "Tracking a moving object with a binary sensor network," in *Proceedings of the 1st international conference on Embedded networked sensor systems*, ser. SenSys '03, 2003, pp. 150–161.

[22] D. Eickstedt and M. Benjamin, "Cooperative target tracking in a distributed autonomous sensor network," in *OCEANS 2006*, September 2006, pp. 1–6.
[23] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, pp. 174–188, 2001.
[24] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, and H. Z. et al., "A line in the sand: A wireless sensor network for target detection, classification,and tracking," *Computer Networks (Elsevier)*, vol. 46, no. 5, pp. 605–634, 2004.
[25] R. M. Taqi, M. Z. Hameed, A. A. Hammad, Y. S. Wha, and K. K. Hyung, "Adaptive yaw rate aware sensor wakeup schemes protocol (a-yap) for target prediction and tracking in sensor networks," *IEICE - Transactions on Communications*, vol. E91-B, no. 11, pp. 3524–3533, 2008.
[26] Y. Zou and K. Chakrabarty, "Distributed mobility management for target tracking in mobile sensor networks," *Mobile Computing, IEEE Transactions on*, vol. 6, no. 8, pp. 872–887, Aug. 2007.
[27] "Kinematics." [Online]. Available: http://en.wikipedia.org/wiki/Kinematics
[28] M. Chu, H. Haussecker, and F. Zhao, "Scalable information-driven sensor querying and routing for ad hoc heterogeneous sensor networks," *International Journal of High-Performance Computing Applications*, vol. 16, no. 3, pp. 293–313, 2002.
[29] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, New York, NY, USA, 2004, pp. 95–107.
[30] J. Hightower and G. Borriello, "Location systems for ubiquitous computing," *IEEE Computer*, vol. 34, no. 8, pp. 57–66, August 2001.
[31] R. Stoleru, J. A. Stankovic, and S. H. Son, "Robust node localization for wireless sensor networks," in *EmNets '07: Proceedings of the 4th workshop on Embedded networked sensors*, 2007, pp. 48–52.
[32] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," *SIGOPS Oper. Syst. Rev.*, pp. 147–163, 2002.
[33] B. Jiang, B. Ravindran, and H. Cho, "Cflood: A constrained flooding protocol for real-time data delivery in wireless sensor networks," in *Proceedings of the 11th International Symposium on Stabilization, Safety, and Security of Distributed Systems*, 2009, pp. 413–427.
[34] "Normal distribution." [Online]. Available: http://en.wikipedia.org/wiki/Normal_distribution
[35] B. Jiang, K. Han, B. Ravindran, and H. Cho, "Energy efficient sleep scheduling based on moving directions in target tracking sensor network," in *IPDPS*, 2008, pp. 1–10.
[36] J. Denga, Y. S. Hanb, W. B. Heinzelmanc, and P. K. Varshney, "Balanced-energy sleep scheduling scheme for high density cluster-based sensor networks," in *Computer Communications: special issue on ASWN04*, vol. 28, 2005, pp. 1631–1642.
[37] J. Liu, J. Liu, J. Reich, P. Cheung, and F. Zhao, "Distributed group management in sensor networks: Algorithms and applications to localization and tracking," in *Telecommunication Systems*, vol. 26, no. 2-4, 2004, pp. 235–251.
[38] M. Athanassoulis, I. Alagiannis, and S. Hadjiefthymiades, "Energy efficiency in wireless sensor networks: A utility-based architecture," in *European Wireless*, 2007.
[39] J. Hill and D. Culler, "Mica: a wireless platform for deeply embedded networks," *Micro, IEEE*, vol. 22, no. 6, pp. 12–24, Nov/Dec 2002.
[40] B. Jiang, "Energy efficient target tracking in wireless sensor networks: Sleep scheduling, particle filtering, and constrained flooding," Dissertation, Virginia Tech, 12 2010.
[41] A. T. Inc., "Dc source with battery emulation," http://www.home.agilent.com/.
[42] X. Jiang, P. Dutta, D. Culler, and I. Stoica, "Micro power meter for energy monitoring of wireless sensor networks at scale," in *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, 2007, pp. 186–195.
[43] C. Margi, V. Petkov, K. Obraczka, and R. Manduchi, "Characterizing energy consumption in a visual sensor network testbed," in *In 2nd International IEEE/Create-Net Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TridentCom 2006)*, 2006.



Bo Jiang obtained his Ph.D. degree in the Department of Electrical and Computer Engineering in Virginia Tech in 2010. His research interests include wireless sensor networks, real-time and distributed systems, and virtualization techniques. He received his B.S. degree and M.S. degree from the Department of Computer Science and Technology in Tsinghua University, China in 2001 and 2004 respectively. Before joining Virginia Tech, he worked as a software engineer in Intel China Research Center for three years, where he conducted development and customization efforts for Linux device drivers on Intel's desktop and mobile platforms. Now he returned to Intel and is working on smart phone platforms as a senior software engineer.

Binoy Ravindran is an Associate Professor in the ECE department at Virginia Tech, Blacksburg, Virginia, USA. His research interests include distributed, real-time, embedded, and networked systems, with a particular focus on robust resource management at various levels of abstraction from OSes to virtual machines to runtimes to middleware, and concomitant programming abstractions. He and his students have published more than 170 papers in this space, and some of his groups results have been transitioned to US Department of Defense programs. Dr. Ravindran is an US Office of Naval Research Faculty Fellow, an ACM Distinguished Speaker, a former IEEE Distinguished Visitor, and an Associate Editor of ACM Transactions on Embedded Computing Systems.

Hyeonjoong Cho is an Associate Professor in the Department of Computer and Information Science at Korea University. His research focuses on real-time systems on various platforms including single/multiprocessors, sensor networks, etc. He is also interested in real-time operating systems, embedded systems, and industrial field bus. Before he joined Korea University in 2009, he worked as a senior researcher in Electronics and Telecommunications Research Institute. He received the PhD degree in computer engineering from Virginia Polytechnic Institute and State University (Virginia Tech) in 2006. He received his M.S. degree in Electronic and Electrical Engineering from Pohang University of Science and Technology, South Korea (1998) and B.S. degree in Electronic Engineering from Kyungpook National University (1996). He had worked for Samsung Electronics as a senior software engineer at Factory Automation research institute before pursuing Ph.D. degree.