# TPSS: A Two-phase Sleep Scheduling Protocol for Object Tracking in Sensor Networks

Qianqian Ren, Jianzhong Li and Hong Gao
School of Computer Science and Technology, Harbin Institute of Technology,Harbin, China
Emails: {qqren,lijzh,honggao}@hit.edu.cn

*Abstract*—**Lifetime maximization is an important factor in the design of sensor networks for object tracking applications. Some techniques of node scheduling have been proposed to reduce energy consumption. By exploiting the redundancy of network coverage, they turn off unnecessary nodes, or make nodes work in turn, which require high nodes density or sacrificing tracking quality. We present TPSS, a two-phase sleep scheduling protocol, which divides the whole tracking procedure into two phases and assigns different scheduling policies at each phase. To balance energy savings and tracking quality, we further optimize the node scheduling protocol in terms of network coverage and nodes state prediction. We evaluate our method in an indoor environment with 36 sensor nodes. Comparing with the existing methods, all the experimental results show that our method has a high performance in term of energy savings and tracking quality.**

## I. Introduction

Object tracking is an important issue of wireless sensor networks. It has been applied in various areas such as disaster predication, emergency response and battlefield surveillance, etc.

Many object tracking protocols have been proposed to support long-term surveillance by using large scale wireless sensor networks [1], [2], [3], [4], [5]. These protocols often consider requirements of fast response to the object and accurate tracking. Since sensor nodes in wireless sensor networks are normally battery powered, optimizing protocols in terms of energy conservation is of critical importance.

In most cases, sensor networks are deployed with a large number of sensor nodes in danger or non-arrival environments, it is unfeasible to replace or recharge power sources. Thus, energy plays a key role in extending the network lifetime. In object tracking applications, sensor nodes collaborate in detecting and tracking the interested target. Generally, the whole tracking procedure contains two phases: one is discovering the appearance of any target, while the other is collecting sensing data and locating the target. Neither of the two phases involve all nodes during the whole system lifetime. That means only the nodes within a local area surrounding the object are required to participate in tracking.

Considering the scenarios of tracking, it is efficient to minimize nodes involved in communicating or processing [3], [4], [6]. The nodes that do not work stay idle. However, sensor nodes also consume energy when idle. It is shown that energy consumption in idle state can not be ignored [2], [6]. It costs sensor nodes nearly 99% of the whole energy when they are ready for the occurrence of any mobile objects in idle state [2].

Some works have been done about applying node scheduling protocols in object tracking applications. These protocols make idle nodes sleep to achieve further power conservation [5], [7], [8], [9]. By exploiting the redundancy of network coverage, they power down unnecessary nodes, or rotate them in work, which require high nodes density or sacrificing tracking quality. In this paper, we propose a two-phase sleep scheduling protocol for object tracking. It assigns different sleep scheduling mechanisms at the corresponding tracking phase.

Initially, all sensor nodes in the monitored area are active. At the first phase, we pick out a set of nodes on duty and make them cycle between awake and sleep, meanwhile have other nodes slept. The nodes on duty are in charge of discovering the object, and awakening necessary sleeping nodes to track the object. At the second phase, partial nodes surrounding the object are wakened to sample data of the object. They turn to sleep again when the object has moved out.

Intuitively, choosing a smaller set of nodes on duty can provide more power savings, while may lead to higher detection delay. Energy savings and tracking quality are two conflict requirements. This paper attempts to balance them and formalize it as the network coverage problem. This problem is known to be NP-hard. We give an approximation solution based on strips division, which has approximation ratio 2 for specific instances.

Besides, we observe that the object does not move randomly, it is highly dependent on its previous movement. If we can predict the appearance area of an object using this dependence, we may awaken sleeping nodes ahead. Most works estimate the existence area of the object by constructing its trajectory model [10], [11], [12], which is too expensive. Alternatively, this paper converts it into the problem of state transitions of nodes. We explore the change trend of their transitions using Markov chain theory.

The contributions of this paper are as follows. First, we present TPSS, a two-phase sleep scheduling protocol in detail, which is the first protocol to provide varied node scheduling mechanisms according to the different tracking phase. Second, we further optimize TPSS considering parameters of network coverage and nodes' state transitions. These optimizations provide better tracking quality with energy conservation guar-

978-1-4244-5113-5/09/$25.00 ©2009 IEEE

antee. Third, we evaluate TPSS through a network with 36 sensor nodes in an indoor environment. Experimental results show that TPSS provides better quality of tracking while reducing the energy consumption. The remainder of this paper is organized as follows. Related work is shown in Section 2. We describe the sleep scheduling protocol in Section 3. Two optimization methods are presented in Section 4. We briefly describe our prototype and present detailed evaluations in Section 5. Finally, we conclude this paper in Section 6.

## II. Related Work

Several research works have been done for energy efficient object tracking in wireless sensor networks [1], [3], [5], [7], [8], [13], [14], [15], [16], [17]. Shrivastava et.al [15], [16], [17] propose methods that track the object with binary sensor model. The output of each binary sensor is only one bit (0 or 1), it can reduce the amount of transmissions. However, this model requires reliable networks, for any loss of packets affects the tracking accuracy evidently. Cricket [13] is an indoor location system, which locates objects using TDOA technique. This system equips costly hardware-ultrasonic, while its directional demand cannot locate for all directions. DCTC [14] is a distributed tracking algorithm using dynamic convoying tree structure for node collaborations. It optimizes the problem by finding a convoy tree sequence with high tree coverage and low energy consumption. ZhaoF et.al [4] propose an information-driven approach, which decides the sensor collaboration on constraints of information, cost and resource consumption. MCTA [3] gives a minimal contour-tracking algorithm to minimize the number of working nodes. Lee J et.al [6] suggest a distributed energy efficient tracking method, which conserves energy by reducing the number of transmitted message.

The previous tracking techniques have considered requirements of energy savings in object tracking applications. Their solutions focus on reducing transmission amounts or the number of nodes participating in work. While the nodes that do not work still consume energy, and these consumptions cannot be ignored for the long-term surveillance.

Some methods have been proposed to apply node scheduling in object tracking systems [5], [7], [8]. They can save energy consumption of networks obviously. VigilNet [5] is a sleeping mechanism based tracking system. It integrates a tripwire service with a sentry and duty cycle scheduling to increase the system lifetime. Cao et.al [7] propose an optimal node sleep scheduling protocol for rare-event detection. This protocol focuses on developing a deterministically rotating sensory coverage with constrains of detection delay. Gui et.al [8] study the soft deployment technique and design an efficient sleep-awake protocol. It wakes up and shuts down sensor nodes with spatial and temporal preciseness.

These node scheduling based methods suppose that the network is covered with certain density. By exploiting the redundancy of nodes coverage, they make redundant nodes sleep, or rotate them in work. This scheme will influence tracking quality in terms of detection delay and tracking precision.

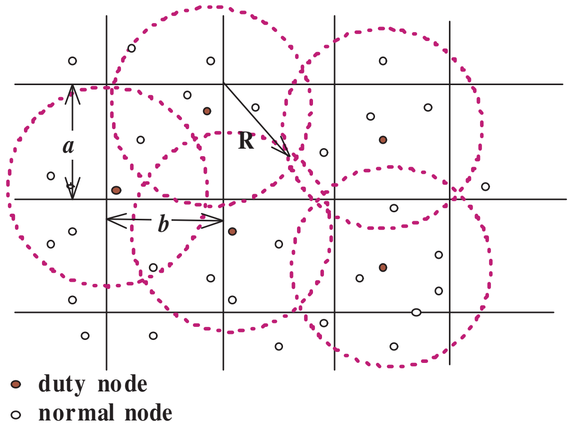


Fig. 1. Network Model

In our approach, we schedule nodes according to the different tracking phase. At the discovering phase, we choose a smaller set of nodes on duty to discover the object and wake up nodes surrounding the object; while in tracking phase, nodes are waken ahead to locate the object. Our method provides not only fast response to the object, but sufficient nodes to perform localization task, meanwhile saves the energy consumption.

## III. Two-Phase Sleep Scheduling Protocol

### A. General Framework of TPSS

This section presents the general framework of TPSS, which is based on the following assumptions:

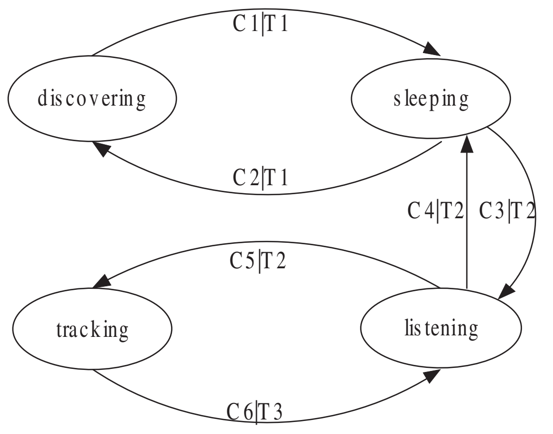
1. An area is covered by a large number of homogeneous sensor nodes with redundant density. The node can detect its own location, via GPS or a localization technique.

2. The sensing area of a node is a circle centered at this node with radius $R$. The sensing area of node $i$ is denoted as $C_i$.

3. The communalization range of a node equals to its sensing range.

For simplify, we assume that a sensing area is divided into small grids. A grid is an $a \times b$ rectangle as shown in Fig. 1. We assign *id*s for grids, and all nodes can identify *id* of the grid they belong to. According to their assignments, we classify nodes into three types: *duty nodes*, *normal nodes* and *influence nodes*. Duty nodes cycle between sleeping and awake state, they are in charge of discovering the object and sending awakening messages to sleeping nodes. Normal nodes sleep for most time, and wake up periodically to detect if there are awakening messages to them. Influence nodes are special normal nodes. They are responsible for sampling sensing data of the object.

We suppose that the duty node follows duty cycle $\gamma = T_{wake}/(T_{wake} + T_{sleep})$, where $T_{wake}$ is waking time and $T_{sleep}$ is sleeping time. The normal node detects the awakening message at the interval $T_{detect}$, referred to **detection period**. TPSS divides the whole tracking procedure into two phases,



Conditions:
C1: after $T_{wake}$ C2: afer $T_{wake}$
C3: afer $T_{detect}$ C4: not receive awakening msg
C5: receive awakening msg C6: finish tracking
Node types:
T1: duty node T2: normal node T3: influence node

Fig. 2. Nodes state transitions in TPSS

and schedules nodes accordingly. At the first phase, we choose one normal node as duty node in per-grid. In order to balance the energy consumption, nodes in the same grid take turns to be duty node. The duty node focuses on discovering the object entering into its area and sending awakening messages to sleeping normal nodes. Normal nodes keep sleeping. Once the object is detected, parts of sleeping normal nodes switch to the scheduling of second phase. At this phase, they wake up and keep active until the object has move out its sensing area, then they can return to the scheduling of the first phase.

## B. Nodes State Transitions

In TPSS, nodes have four states: *discovering*, *sleeping*, *listening* and *tracking*. The duty node cycles between discovering and sleeping states. The normal node is in sleeping state mostly, and periodically turns into listening state to detect if there is an awakening message for it. When the normal node receives an awakening message, it enters tracking state, and becomes an influence node. A state transitions diagram is shown in Fig. 2.

A duty node cycles between discovering and sleeping state with duty cycle $\gamma$. When in discovering state, the duty node detects if there is an object entering into its sensing coverage. If so, it sends an awakening message to normal nodes in its grid. The awakening message is a tuple of message type and grid *id*.

A normal node stays in sleeping state at most time and turns into listening state at intervals $T_{detect}$. When in listening state, it detects if there is an awakening message for it. If so, it wakes up and turns into tracking state. Meanwhile, the normal node becomes an influence node.

The influence node in tracking collects sensing data of the object and sends it to its duty node for further process. It performs the task continuously until the object has moved out of its sensing area, then it finishes sensing and turns to listening state. The influence node entering listening state returns to the original sleeping plan and turns to a normal node again.

## C. Tracking performance analysis

Here we consider three performance metrics, which are crucial to estimate quality of the tracking system. These metrics are defined and analyzed as follows.

1. **Detection probability**: it is the probability that at least one duty node detects the object. It is assumed that an active sensor node can detect the object with probability of $\eta$ $(0 \leq \eta \leq 1)$ when this object is in its coverage area. $\eta$ is determined by physical attributes of the sensor, we can learn it from historic data [21]. For the purpose of a generalized analysis, we assume that all sensors have the same $\eta$.

If a duty node can discover the object, there must be an overlapping between its sensing area and the trace of the object, meanwhile the duty node is in discovering state. Thus, the detection probability relates to the interval that the object stays in sensing area of the duty node and its duty cycle. The probability that one duty node detects the object can be represented as:

$$P_s = \begin{cases} \frac{\Delta t}{T_{sleep}}\eta & \text{if } \Delta t \leq T_{sleep} \\ \eta & \text{else} \end{cases} \tag{1}$$

Where $\Delta t$ is the duration that the object stays in sensing area of the duty node. Shrivastava.et.al [15] has presented its estimation method in detail.

When $\Delta t > T_{sleep}$, the object is detected with the probability of $\eta$, so $P_s$ only depends on physical attributes of sensors.

We suppose that any point in the environment is covered by *m* duty nodes on average. The detection probability is:

$$\begin{aligned} P_{detection} &= 1-(1-P_s)^m \\ &= \begin{cases} 1-(1-\frac{\Delta t}{T_{sleep}}\eta)^m & \text{if } \Delta t \leq T_{sleep} \\ 1-(1-\eta)^m & \text{else} \end{cases} \end{aligned} \tag{2}$$

Formula (2) shows that $P_{detection}$ is determined by $\Delta t$, *m* and $T_{sleep}$. $\Delta t$ is influenced by movement direction and speed of the moving object and nodes' sening range *R* [15]. When fixing $\Delta t$ and *R*, we can improve $P_{detection}$ by increasing *m* or decreasing $T_{sleep}$. However, it leads to more energy consumptions. We explore this conflict in section 4.1, where we attempt to give an algorithm to optimize the problem of determining duty nodes. It releases above conflict from the aspect of the distribution of duty nodes.

2. **Detection delay**: It is defined as the time elapsed between object occurrence at any point and its detection by a duty node [7]. We suppose that a sensor node takes time $T_{start}$ for changing its state to work from sleep, after that, it uses $T_{sample}$ to get sensing data. Then, detection delay can be represented as:

$$T_{detection\_delay} = \alpha T_{sleep} + T_{start} + T_{sample} (0 \leq \alpha \leq 1) \tag{3}$$

and

$$\alpha = (1-\gamma)\gamma^m \tag{4}$$

*m* is defied as formula (2), duty cycle $\gamma$ can be viewed as the probability that a duty node stays sleeping, then $\alpha$ is the probability that none of *m* duty nodes stay in discovering at the time that the object appearing.

In formula (3), $T_{start}$ and $T_{sample}$ are dependent on hardware capabilities. Therefore, detection delay is only decided by $\gamma$ and *m*. Evaluation results are similar to the case of detection probability.

3. **Tracking delay**: It is defined as the time elapsed between object appearance at a point and its location response to the base station. After getting sample data, it takes $T_{process}$ to process information, and $T_{routes}$ to transmit results to the base station. Thus, tracking delay can be presented as:

$$\begin{aligned} T_{tracking\_delay} &= T_{detection\_delay} + \beta T_{detect} + T_{start} \\ &+ T_{sample} + T_{process} + T_{route} \quad (0 \le \beta \le 1) \end{aligned} \tag{5}$$

In formula (5), $T_{start}$, $T_{sample}$ and $T_{process}$ depend on the hardware capabilities, while $T_{route}$ is decided by particular routing algorithm, which is not considered in our algorithm. So the tracking delay is affected by $\beta T_{detect}$. If any point in the environment is covered by *k* normal nodes on average, then $\beta$ is the probability that none of *k* normal nodes are in listening state.

From formula (5), we know that $T_{detection\_delay}$ and $\beta T_{detect}$ are key factors that affect tracking delay. If we can predict nodes needed to be awakened up, and awaken them in advance, $\beta T_{detect}$ can be decreased to zero. In section 4.2, we explore the change trend of nodes state transitions, and predict inluence nodes based on this rule.

## IV. Optimizing sleep Scheduling

Previous sections have introduced a node sleeping scheduling scheme to conserve energy in the tracking application. We now will optimize it from aspects of duty nodes coverage and predicating states of normal nodes.

### A. Determining Duty Nodes

With the guarantee of tracking quality and energy savings, we try to find out the minimum number of duty nodes, which can cover the monitor field. This problem can be view as an issue of duty nodes coverage. As this problem is NP-complete, we will give an approximation solution.

Several works introduce strips division algorithm to solve the plane coverage problems [18], [19], [20] . Similarly, our solution divides the whole network into multiple strips. Unlike to their covering points within each strip, we adopt greedy heuristic method to divide each strip into minimum number of grids, moreover each one is covered by at least one duty node.

The procedure of determining duty nodes is composed of two phases: 1) Divide the whole network into equal width strips; 2) Decide local duty nodes within each strip.

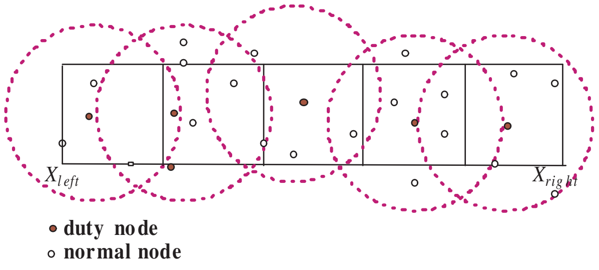


Fig. 3. An example of strip division algorithm

**Algorithm 1**: Duty-nodes-decision

```
Input: (1) S[1...n], (2) I[1...n], and (3) $x_{left}$ and $x_{right}$
Output: A set of duty nodes
1: while $x_{left} < x_{right}$ do
2:     for each node S[i]
3:     if the distance from $x_{left}$ to I[i].x is the greatest and S[i]can cover
       the grid with x-coordinates $x_{left}$ and I[i].x then
4:         set S[i] as a duty node
           $x_{left}$ = I[i].x
5: Output the set of duty nodes
```

*1) Network division:* In the first phase, the whole network is divided into strips with width of $\lambda$R, as shown in Fig.3. Where $\lambda$ is referred to **division factor**, it affects not only the results of strips division, but the whole networks coverage. We will investigate its effect experimentally in section 5.

*2) Duty nodes decision:* We define the sub-problem of this phase as follows.

**Problem definition:** Given a set of sensor nodes distributed in the strip, find the smallest set of them as duty nodes such that the strip is covered by these duty nodes.

We solve the above problem by dividing the strip into multiple grids. The goal is to make the number of grids minimum, meanwhile each grid is covered by at least one duty node. Given a fixed width strip, it is desirable that the length of each grid is as great as possible. Algorithm 1 presents the process of duty node decision.

In this algorithm, array *S*[1...*n*] lists the sensor nodes in a given strip, ordered by their ascending *x*-coordinates. *S*[*i*].*x* denotes *x*-coordinate of node *S*[*i*]. *I*[1...*n*] records the intersection of nodes in strip *P* and its boundary[1]. For example, *I*[i] is the intersection of right half of $C_{S[i]}$ and strip *P* . *I*[*i*].*x* is *x*-coordinate of this intersection. $x_{left}$ is *x*-coordinate of the leftmost side of strip *P*, while $x_{right}$ is its rightmost side *x*-coordinate, as shown in 3.

As algorithm 1 illustrated, from the leftmost side of *P*, we would like choose a node that can cover the grid with greatest length. The chosen node is a duty node of the gird, of which *x*-coordinates are $x_{left}$ and *I*[*i*].*x*, respectively. After that, this grid is cut off from *P*, we recursively solve the rest part of *P* as the same way, until the whole strip is covered by duty nodes. Fig.3 shows an example of the strip division.

[1]We only consider the intersections of right half of nodes's sensing area and *P*. If the *x*-coordinate values of two intersections are different, get the smaller one.

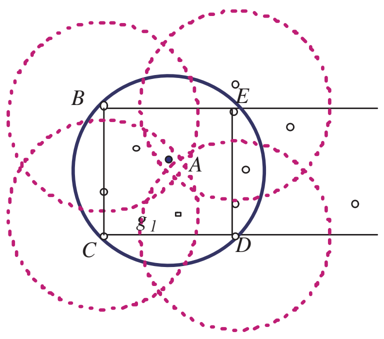


Fig. 4. An example of theorem 1. The optimal solution is given by node *A*, while the worst case selects four nodes

In this algorithm, sorting nodes by their *x*-coordinates takes time O(nlog*n*), and computing *I*[1...*n*] takes time O($n^2$). Therefore, the complexity of Algorithm 1 is O($n^2$).

## B. Approximation Ratios

Make $\Lambda$ denote Algorithm 1, and *OPT* is an optimal coverage of the strip. The approximation factor is the number of required duty nodes in the strip covering, denoted as $|\Lambda|$ and $|OPT|$, respectively.

For a given strip *P*, we use $\{g_1, g_2, \dots, g_{|OPT|}\}$ and $\{a_1, a_2, \dots, a_{|\Lambda|}\}$ to present the division by *OPT* and $\Lambda$, respectively, where $g_i$ and $a_i$ are *i*th grid in strip *P*.

*Theorem 1:* When $1 \le \lambda \le \sqrt{2}$, $\Lambda$ is a 4- approximation algorithm for the problem.

**Proof.** We prove this theorem by inducting the number of grids *m* in every strip.

For *m*=1. The gird $g_1$ can be covered by one duty node (such as node *A*), as shown in Fig.4. The disk denotes the sensing area of a node. For the worst case of algorithm $\Lambda$ , the chosen duty nodes are on the vertexes of the grid. In this situation, four duty nodes (such as nodes *B*, *C*, *D*, *E*) are sufficient to cover this area.

Now suppose that $\Lambda$ uses $m$ duty nodes to cover the area composed of $\{g_1, g_2, \dots, g_{|OPT|-1}\}$, $m$ is not more than $4(|OPT| - 1)$. For grid $g_{|OPT|}$, $\Lambda$ can cover it using four duty nodes at most. Thus, $\Lambda$ can cover the whole strip with *m*+4 duty nodes, where

$m + 4 \le 4(|OPT| - 1) + 4 = 4|OPT|$. □

*Theorem 2:* When $\sqrt{2}/2 \le \lambda \le 1$, $\Lambda$ is a 2- approximation algorithm for the problem.

**Proof.** The proof of theorem 2 is similar to theorem 1, we omit it here. Fig.5 shows an example of theorem 2. □

## C. Predicting influence nodes

In previous section, we have shown that predicting nodes in charge of tracking (that is influence nodes) in advance can reduce tracking delay efficiently. In this section, we term it as the problem of estimating nodes state, and use Markov chain theory to solve it.

Each normal node must be in one of two states: influence node or non-influence node. Without loss of generality, we use $X_t$ (*t*= 1, 2,...) to denote the state of a normal node at

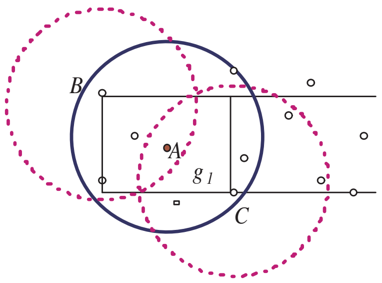


Fig. 5. An example of theorem 2. The optimal solution is given by node *A*, while the worst case selects two nodes

time *t*, and use *i* and *j* to denote two states of a normal node. We suppose that whenever the node is in state *i*, there is a probability $p_{ij}$ that it will next be in state j. That is:

$$P\{X_{t+1} = j | X_t = i, X_{t-1}, \dots, X_1\} = P_{ij} \tag{6}$$

We know that the node state $X_t$ of time *t*, depends only on the state $X_{t-1}$ of time *t*-1 and is independent of the past states. Thus, formula (6) can be represented again as:

$$P\{X_{t+1} = j | X_t = i\} = P_{ij} \tag{7}$$

Suppose that if a node is an influence node at time step *t*, then it will maintain influence node state at time step *t*+1 with probability $\alpha$; and if it is a non-influence node at time step *t*, then it will turn to influence node state at time step *t*+1 with probability $\beta$ .

If we say that the process is in state 0 when a node is an influence node and state 1 when it is a non-influence node, then the proceeding is a two-state Markov chain whose transition probability are given by

$$\mathbf{P} = \begin{bmatrix} \alpha & 1-\alpha \\ \beta & 1-\beta \end{bmatrix} \tag{8}$$

We assume that the target's location is ($x_t$, $y_t$) at time step *t*, its trajectory from time step *t* to *t*+1 must be in a circle centered of ($x_t$, $y_t$), with radius of the value of object moving speed. We denote this circle as $D_{t+1}$.

If a node *i* is an influence node at time step *t*+1, then its sensing area must overlap $D_{t+1}$. The overlapping area between $C_i$ and $D_{t+1}$ is denoted as $S_i$. The larger the area of $S_{t+1}$, the more possible node *i* becomes an influence node. Thus $\alpha$ and $\beta$ can be expressed as:

$$\alpha = \frac{\omega_1 S_i}{\pi * R * R}, \quad \beta = \frac{\omega_2 S_i}{\pi * R * R}$$

$\omega_1$ and $\omega_2$ are two weights, we can get them from experiences. They are decided by the target's moving speed, direction and nodes' sensing area, etc.

According to [22], the limiting probabilities $\pi_0$ and $\pi_1$ are given by

$$\begin{aligned} \pi_0 &= \alpha\pi_0 + \beta\pi_1 \\ \pi_1 &= (1-\alpha)\pi_0 + (1-\beta)\pi_1 \\ \pi_0 + \pi_1 &= 1 \end{aligned}$$

which yields that

$$\pi_0 = \frac{\beta}{1+\beta-\alpha}, \quad \pi_1 = \frac{(1-\alpha)}{1+\beta-\alpha} \tag{9}$$

With formula (9), we can obtain nodes satisfying the condition $\pi_0 \geq P_{Threshold}$. $P_{Threshold}$ is defined by users. These nodes are influence nodes. The procedure is summarized in Algorithm 2.

**Algorithm 2**: InfluenceNodes-prediction

```
Input: (1) transition matrix at time step $t$ , (2) $P_{Threshold}$, (3) moving
speed of the object, and (4) target's location ($x_t$, $y_t$) at time step $t$
Output: A set of influence nodes
 1: calculate a circle $D_{t+1}$ centered of ($x_t$, $y_t$), with the radius of the
    value of object moving speed;
 2: for each node $i$ in $I = \{i|C_i \cap D_{t+1} \neq \varnothing\}$ do
 3:     compute $\pi_0$;
 4:     if $\pi_0 \geq P_{Threshold}$ then
 5:         remove $i$ from $I$;
 6: Output $I$;
```

In Algorithm 2, the complexity of step 2-3 is O($N^2$), so the time complexity of the algorithm is O($N^2$).

## V. Experiments and Evaluations

This section presents a careful evaluation of TPSS. Firstly, we explore the impact of different parameters on the system performance, and then we compare our approach with other energy efficient tracking protocols. We consider parameters of detection delay, tracking delay, detection probability and energy savings ratio. We investigate the effect of some key factors including division factor, duty cycle and detection period on above system parameters. Besides, we implement VigilNet [5] and MCTA [3] in our network, and compare them with TPSS. These two protocols are lately representative ones in object tracking applications. VigilNet also applies node scheduling mechanisms in its surveillance applications and MCTA conserves energy by reducing work nodes in the tracking procedure.

### A. Prototype Setup

A small experiment network with 36 sensor nodes is constructed. We implement TPSS in TinyOS 2.x operating system, programmed in NesC language [23]. Each node features a MSP430 processor and a nRF905 wireless communication component. We adjust the work frequency and power of radio component, to make each sensor have a sensing range about 2 meters. We use an electronic car as the mobile object. The object equips a radio component, which can send radio signals continuously. A sensor node can detect the signal and discover the object when the object moves into its radio area. The mobile object can move along any direction in a straight line.

### B. Detection Delay

In this section, we study average detection delay, which is the average time elapsed between object appearance in the area and its detection by one duty node. The results of average detection delay for varying division factors and duty cycles are shown in Fig.6. When we set division factor as 1.00, detection delay is lowest; and while it is set as 1.25, detection delay is highest. That is because division factor determines not only the number of strips and grids in each strip, but also the coverage degree of duty nodes. In our experiments, when division factor is set as 1.25, the whole network is divided into minimum grids, while when it is set as 1.00, the number of grids is maximum. For the latter case, the network is covered by maximum duty nodes, thus the object has more chances to be discovered. It is also indicated according to the figure that, duty cycle has obvious influences on detection delay. The larger the duty cycle is, the shorter average detection delay we get. The reason is that increasing duty cycle makes nodes have less sleeping time, so they have more time to work and discover the object.

### C. Tracking Delay

This section studies average tracking delay, which is the average time elapsed between object appearance in the area and its response to the base station. Fig.7 illustrates the average tracking delay of different division factors and duty cycles. Duty cycle and division factor are also key factors that influence tracking delay.

Fig.8 presents the influence of detection period on tracking delay varying duty cycle, when setting division factor as 0.8. We observe that tracking delay increases clearly as the increase of detection period.

### D. Detection Probability

According to previous sections, division factor and duty cycle are key factors decides detection probability. Fig.9 illustrates the results. As expected, the detection probability is proportional to duty cycle. Similar to detection delay, the case of which division factor is 1.00 performs best, and the case of which division factor is 1.25 performs worst. As shown in this figure, detection probability reaches to nearly 100% with proper parameters configuration, and no less than 85% in worst-case. It illustrates that our sleeping mechanism does not have evident influence on the system.

### E. Energy Savings Ratio

We define the energy savings ratio as the ratio of energy savings of the algorithm over the normal tracking system without node scheduling mechanisms. We consider the sum of energy consumption, including sampling sensor data, communication by wireless component and staying in working/sleeping state. We compute the consumed energy as literature [5]. Table I presents the power consumed by sensor nodes when they are in different states.

TABLE I
Energy Consumed by Nodes in different states

| Node state | Power consumption |
|---|---|
| Sleeping | $42\mu W$ |
| Listening | $49.449mW$ |
| Discovering | $71.45mW$ |
| Tracking | 70.01mW |

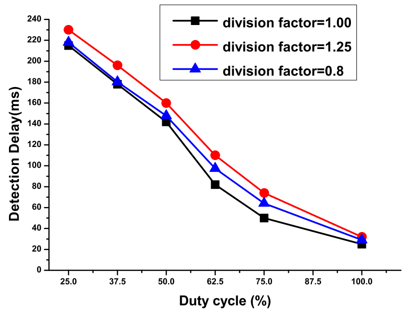


Fig. 6. Impact of division factor and duty cycle on detection delay

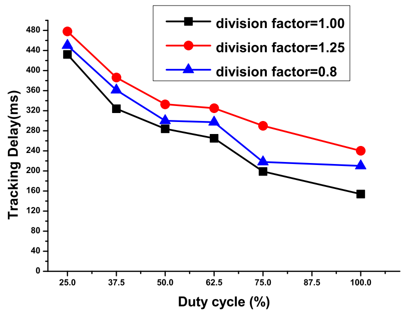


Fig. 7. Impact of division factor and duty cycle on tracking delay

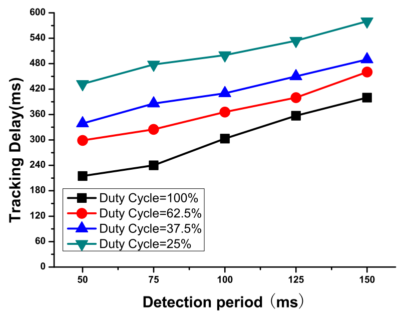


Fig. 8. Impact of duty cycle and detection period on tracking delay

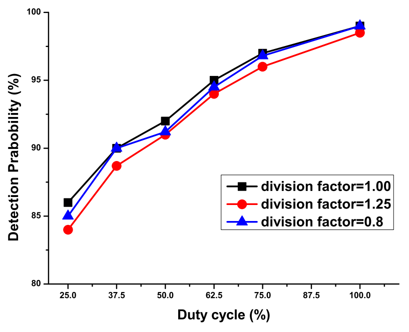


Fig. 9. Impact of division factor and duty cycle on detection probability

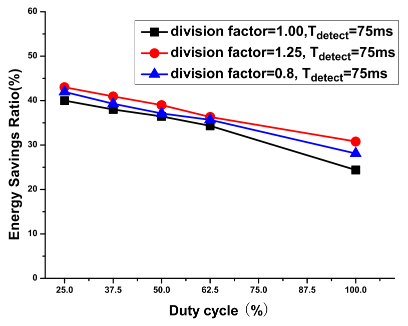


Fig. 10. Impact of division factor and duty cycle on energy savings ratio

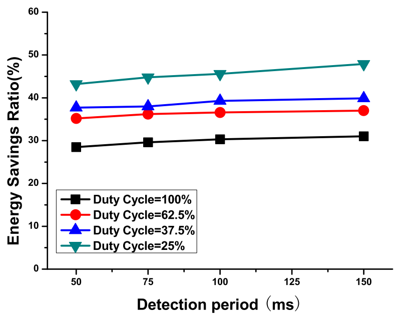


Fig. 11. Impact of duty cycle and detection period on energy savings ratio

Fig.10 and Fig.11 depict the impact of division factor, duty cycle and detection period on energy savings. Obviously, TPSS conserves much energy consumption. In addition, this energy conservation is significantly affected by duty cycle and division factor. Although the number of duty nodes is smaller compared to the total number of sensors, their node scheduling scheme makes them cost much more resource than normal nodes do.

Fig.10 shows that energy savings decreases linearly as the increase of duty cycle. This because the increase of duty cycle leads to longer work time of duty nodes, and more duty nodes detecting the object, thus more information transmitted in networks. Theoretically, the total energy savings should be proportional to the sleeping time. However, nodes state converting and sending awakening message cause to extra energy cost.

Fig.10 also shows the influence of division factor on energy savings, when detection period is fixed. When duty cycle is larger than 50%, the case of which division factor is 1.25 obtains larger energy savings than others do, while when duty cycle is less than 50%, this difference is not obvious.

Fig.11 plots the influence of detection period on energy

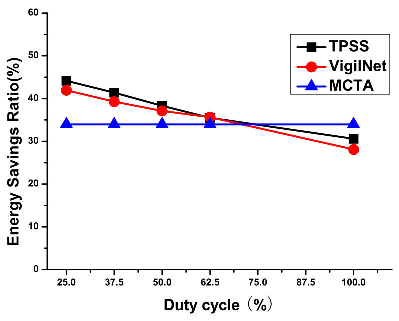


Fig. 12. Energy savings ratio of TPSS, MCTA and VigilNet algorithms

savings varying duty cycle, when division factor is 0.8. It is shown again that duty cycle influences energy consumption more obviously. As the increase of detection period, energy savings increase slightly. For increasing detection period leads to less times of detecting, while this cost is limited compared to others.

Fig.12 compares the energy savings of TPSS with other excellent energy efficient tracking algorithms such as VigilNet and MCTA, when other system parameters including nodes work period and routing, etc. are the same. We observe that, when the duty cycle is no more than 75%, the performances of TPSS and VigilNet are better, that is because they can take good advantage of node scheduling mechanism. However, as the increase of duty cycle, this advantage becomes weaker. MCTA, which is based on minimizing nodes that participate in communication, is not influenced by duty cycle, so performs better when duty cycle is higher.

## VI. Conclusion

In this paper, we present TPSS, a two-phase sleep scheduling protocol that reduces the whole network energy consumption, as well as maintains high tracking quality. To ensure that the protocol exhibits better performance, we optimize it from the aspects of duty nodes decision and nodes state prediction.

We have built a network of 36 sensor nodes in an indoor environment to validate the proposed approach. The results show that TPSS can obtain high energy savings ratio with acceptable detection delay, tracking delay and detection probability.

## Acknowledgment

This work was supported in part by the National Grand Fundamental Research 973 Program of China (Grand No. 006CB303000), the Key Program of National Natural Science Foundation of China (Grant No. 60533110), the NSF of China (Grant No. 60773063 and 60703012),and the NSFC-RGC of China (Grant No. 60831160525).

## References

[1] M. Ding and X. Z. Cheng, *Fault-Tolerant Target Tracking in Sensor Networks*. Proc. 10th ACM international symposium on Mobile ad hoc networking and computing (MOBIHOC '09), pp. 125-134, 2009.

[2] T. He, P. A. 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*. ACM Transaction on Embedded Computing System (TECS), 2007.

[3] J. Jeong, T. Hwang, T. He and D. Du. *MCTA: Target Tracking Algorithm based on Minimal Contour in Wireless Sensor Networks*. Proc. 26th IEEE International Conference on Computer Communications (Infocom '07), pp. 2371-2375,2007.

[4] F. Zhao,J. Shin and J. Reich, *Information-driven dynamic sensor collaboration for tracking applications*. IEEE Signal Processing Magazine, pp. 61-72, 2002.

[5] T. He, P. Vicaire, T. Yan, Q. Cao, G. Zhou, L. Gu, , L. Luo, R. Stoleru, J. A.Stankovic and T. Abdelzaher, *Achieving Long-Term Surveillance in VigilNet*. Proc. 25th IEEE International Conference on Computer Communications(Infocom '06), 2006.

[6] Y. Xu, J.vHeidemann and D. Estrin, *Geography-informed Energy Conservation for Ad-hoc Routing*. Proc. 7th Annual ACM/IEEE International Conference Mobile Computer and Network, pp. 70-84, 2001.

[7] Q. Cao, T. Abdelzaher, T. He and J. Stankovic, *Towards Optimal Sleep Scheduling in Sensor Networks for Rare-Event Detection*. Proc. 4th international symposium on Information processing in sensor networks(IPSN '05), 2005.

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

[9] P. Dutta, M.vGrimmer, A.vArora, S. Bibyk and D. Culler, *Design of a Wireless Sensor Network Platform for Detecting Rare, Random, and Ephemeral Events*, Proc. 4th international symposium on Information processing in sensor networks(IPSN '05), 2005.

[10] Ashwin D'Costa and Akbar Sayeed, *Collaborative signal processing for distributed classification in sensor networks*. Proc. 2nd International Workshop on Information Processing in Sensor Networks (IPSN '03), pp. 193-208, 2003.

[11] D. Li, K. Wong, Y. Hu and A. Sayeed, *Detection, classification and tracking of targets in distributed sensor networks*. IEEE Signal Processing Magazine, vol. 19, no. 2, pp. 17-29, 2002.

[12] D. McErlean and S. Narayanan, *Distributed detection and tracking in sensor networks*. Proc. 36th Asilomar Conference on Signals, Systems and Computers, pp. 1174-1178, 2002.

[13] Nissanka Bodhi Priyantha. *The Cricket Indoor Location System*. PhD Thesis, Massachusetts Institute of Technology, June 2005.

[14] W. S. Zhang and G. H. Cao, *DCTC: Dynamic convoy tree-based collaboration for target tracking in sensor networks*. IEEE Transactions on Wireless Communication, pp. 1689- 1701, 2004.

[15] N. Shrivastava, R. Mudumbai, U. Madhow, and S. Suri, *Target Tracking with Binary Proximity Sensors: Fundamental Limits, Minimal Description, and Algorith*. Proc. 4th international conference on Embedded networked sensor systems(Sensys '06), pp. 251-264, 2006.

[16] W. Kim, K. Mechitov, J. Y. Choi and S. K. Ham, *On Tracking Objects with Binary Proximity Sensors*. Proc. 4th International Symposium on Information Processing in Sensor Networks (IPSN '05), pp. 301-308, 2005.

[17] J. Singh, U. Madhow, R. Kumar, S. Suri and R. Cagley, *Tracking Multiple Targets Using Binary Proximity Sensors*. Proc. 6th International Symposium on Information Processing in Sensor Networks (IPSN '07), pp. 529-538, 2007.

[18] A. Srinivas, G. Zussman and E. Modiano, *Mobile backbone networks-Construction and maintenance*, Proc. 7th ACM international symposium on Mobile ad hoc networking and computing (Mobihoc' 06), pp. 166-177, 2006.

[19] D. S. Hochbaum and W. Maass. *Approximation schemes for covering and packing problems in image processing and VLSI*. Journal of the ACM(JACM), 32(1):130-136, 1985.

[20] T. Gonzalez. *Covering a set of points in multidimensional space*. Proc. Letters, 40(4):181-188, 1991.

[21] S. Oh and S. Sastry, *Tracking on a Graph*. Proc. 4th international symposium on Information processing in sensor networks(IPSN '05), 2005.

[22] S. M. Ross. *Introduction to Probability Models*, 9rd ed. Academic Press, 2006.

[23] UC Berkeley. *TinyOS: an open-source operating system designed for wireless embedded sensor networks*. http://www.tinyos.net/. UC Berkeley, 2004.