Energy Efficient Sleep Scheduling based on Moving Directions in Target Tracking Sensor Network

Bo Jiang*, Kai Han*, Binoy Ravindran*, and Hyeonjoong Cho‡

*ECE Dept., Virginia Tech
Blacksburg, VA 24061, USA
{bjiang,khan05,binoy}@vt.edu

‡ETRI
Daejoen, South Korea
raycho@etri.re.kr

Abstract

This paper presents a target direction-based sleep scheduling algorithm (TDSS) to enhance the energy efficiency for a mobile target tracking surveillance sensor network. TDSS combines the working node reducing efforts and the sleep scheduling, achieves the energy efficiency but suffers little performance loss. For the sleep scheduling, we consider the target's moving direction when defining the tracking subarea to imitate the actual object's motion more likely. Two target moving direction probability distributions are discussed for the sleep scheduling: normal distribution and linear distribution. We compare their performance with the legacy circle-based proactively waking up scheme (Circle) and a working node reducing algorithm - MCTA. The evaluation result shows that TDSS can achieve much better energy efficiency but with little loss on the detection probability and the detection delay.

1. Introduction

Mobile target tracking is one of the most typical applications of the wireless sensor network (WSN), where "tracking" means that when observing a target, a node not only needs to report up to the sink node but also has to make sure to keep tracking without losing. Proactively waking up the neighboring nodes is a commonly used approach for keeping tracking [4, 13].

Energy efficiency is one of the most critical issues for a sensor network, since a node's power supply is hard to be replaced once deployed. The energy efficiency is a must-have attribute in almost every WSN research area including the mobile target tracking. Usually there are many approaches to achieve it, such as scheduling the nodes' sleep pattern [3], optimizing the sensing coverage or the network topology [11], and imposing control on the RF radio [9] etc. This paper will focus on the sleep scheduling approach.

In the past sleep scheduling works, some do not consider the target's movement at all [6], and others may schedule the sleep pattern based only on the distance of a node from the target's instant position: the closer the node is from the target, the lighter its sleep level should be. In another word, they consider the magnitude of a target's velocity only. Such a sleep scheduling algorithm or protocol is usually called "circle-based scheme" or "legacy scheme" [5, 13]. With this scheme, all the nodes in a circle will take the same sleep pattern without distinguishing the difference on moving directions.

However, a target in the real world usually has its "purpose", e.g. intruding towards a specific goal. And a target, i.e. a vehicle, will be restrained by physics rules. For example, a vehicle engine's maximum power output is fixed, thus the accelerator will decrease as the velocity increases. Therefore the target's movement will follow some specific moving model instead of being totally random. If the target's kinematics is taken into account, not all the nodes in a circle need to be awakened to get prepared, thus the consumed energy could be reduced accordingly. [9] made an initial attempt on this way by introducing a minimal contour tracking algorithm (MCTA) to cut down the working node quantity. But MCTA still keeps all the nodes in the contour alive without sleep scheduling.

This paper presents a target direction-based sleep scheduling algorithm (TDSS) to enhance the energy efficiency for a mobile target tracking surveillance sensor network. Also we will introduce a direction-based tracking contour deciding mechanism. Compared with MCTA, this mechanism is proven to provide a better tracking performance with its adaptivity for the actual motion model. Our contributions include 1) the modeling of the target's movement, especially the moving direction based on reasonable assumptions; and 2) a sleep scheduling algorithm based on this target movement model to enhance the energy efficiency. The results from our analysis and evaluation show that compared with the circle-based sleep scheduling and the MCTA algorithm, our sleep scheduling algorithm will exceed on the energy efficiency by $(10\% - 200\%)$.

The rest of this paper is organized as follows: Section 2 will describe our assumptions, the target movement model, the network architecture model, the sleep level model and our objectives; Section 3 presents our core algorithm – TDSS; Section 4 gives the analysis on the energy consumption and the tracking performance; Section 5 provides the evaluation result and the contributions of our TDSS algorithm compared with the legacy circle-based algorithm and MCTA; at last Section 6 discusses our future work.

2. Models and Objectives

Given that we are focused on the sleep scheduling algorithm, we need to clarify some assumptions for those non-focus features and models.

2.1. Assumptions

We would like to make the following assumptions before diving into our algorithm design:

- Coverage – The target tracking field is fully covered by the nodes' sensing radio ranges.
- Node location – The sensor nodes' location is known a priori via GPS [7] or some other algorithms [10].

- Architecture – All the nodes are homogeneous, and the sensor network is in a flat architecture. Also we assume the tracking field is flat, thus we use a 2D mesh model to describe the network.
- Sink node – All the tracking behaviors are completed locally around the target without the sink node's intervention.
- Target positioning – The sensor nodes can find out a target's position, instant velocity magnitude and direction (either by sensing or calculating) [1,12].

2.2. Target Movement Model

In the real world, the target will in general move towards a certain direction following certain physics rules, for example with a reasonable velocity scope. Here we present a movement model taking into account the moving direction to imitate the actual object motion.

First, for the target's velocity magnitude, we assume that it varies within a scope $[0, v_{max}]$. To simplify the algorithm, we restrain the target speed below an upper threshold so that the proactively waking up action can be accomplished with one hop of communication.

Then for the target's velocity direction, we design two model options based on the assumption that the target has a higher probability to keep the current direction than to change to another, and turning around (making a 180^o U-turn) has the least probability. Figure 1 shows the probability that the target moves to different moving directions in a 2D tracking area, where v is the instant velocity of the target, and the area of the ellipse-like contour is 1. Figure 2 are the probability density functions of the two model options: the normal distribution model and the linear model, where $\theta \in (-\pi, \pi]$ and the direction of $\theta = 0$ is the instant velocity direction at the current time point.

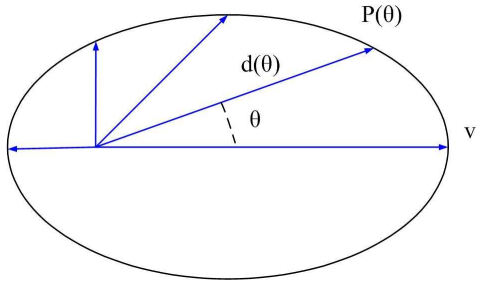

Figure 1. Target Moving Probability on Different Directions

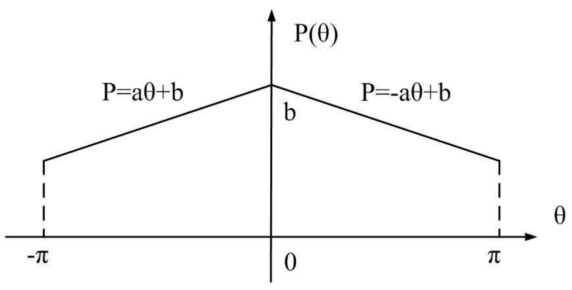

Figure 2. Linear Probability Model

2.2.1 Normal Distribution Model

For the normal distribution option, we set $u = 0$ to guarantee the moving probability on the current direction to be the highest. Thus the probability of moving to the direction θ is described as

$$p(\theta) = \frac{1}{\sigma\sqrt{2\pi}} \exp{(-\frac{\theta^2}{2\sigma^2})}$$

Since the turning angle of a target especially a vehicle cannot be very big when it is moving with a high speed (otherwise it may turn over), the probability should be more concentrative on the current direction as the moving speed increases. Accordingly the variance σ^2 should decrease as the speed increases. Here we set a linear mapping from the target's instant speed v to the variance σ^2 as $\sigma^2 = av + b$. Then

$$p(\theta) = \frac{1}{\sqrt{2\pi(av+b)}} \exp\left(-\frac{\theta^2}{2(av+b)}\right) \tag{1}$$

Sure that this is only an approximate model, since with this normal distribution model, the total probability on all the directions within $(-\pi, \pi]$ is less than 1. But the probability outside of the interval can be considered as for the case that the target stops where it is.

2.2.2 Linear Model

In the linear model, the probability decreases on the side directions linearly and reaches the minimum value at the direction of $\theta = \pi$. We use the following equation to describe it,

$$p = \begin{cases} a\theta + b, \theta \in (-\pi, 0] \\ -a\theta + b, \theta \in (0, -\pi] \end{cases} \tag{2}$$

Since the total probability is 1, we have

$$2\int_0^{\pi} (-ax + b)dx = 1$$

Also the probability of turning around should be no less than 0, we have $p(\theta = \pi) = -a\pi + b \geq 0$. Then we can give out a scope for a and b, in which we use $a = (\pi - 2)/(2\pi^2)$ and $b = 1/4$ as the example.

2.3. Network Architecture Model

Proactively waking up the neighbor nodes requires a root node to broadcast the alert message, the target's position and the velocity info. All the proactively awakened nodes and the root node together form a tracking subarea to keep the tracking action. In fact the circle functions just as such kind of "subarea" in the circle-based sleep scheduling algorithm, but in this paper the subarea is not a circle. Instead both the subarea's shape and its size are decided by the target's movement model.

Let $d(\theta)$ be the radius from the root node to the edge of the subarea at the direction θ (as Figure 1), where θ is the included angle of the target-root node line and the instant velocity. Setup a linear mapping from the maximum directional probability $p(\theta = 0)$ to the subarea's maximum radius $d(\theta = 0)$. On all the other directions, the radius can be given from $d(\theta = 0)$ with proportion. Refer to Equation 3 for this mapping relation.

$$\begin{cases} d(\theta = 0) = m \cdot v \cdot p(\theta = 0) + n \\ d(\theta) = \frac{p(\theta)}{p(\theta=0)} d(\theta = 0) \end{cases} \tag{3}$$

As stated above, the proactively waking up action should complete with one hop of communication. We have $d(\theta = 0)|_{v=v_{max}} \leq d_{com}$, where d_{com} is the the sensor node's RF communication range. Let $d(\theta = 0)|_{v=v_{max}} = d_{com}$, and $d(\theta = 0)|_{v=v_{min}} = d_{min}$, where v_{min} and d_{min} are the assumed minimum speed and minimum subarea radius respectively. Then for each target movement model, we can give out the parameter m and n for Equation 3.

Then the tracking subarea size and its shape will be described with the following equations,

Normal distribution model

$$d(\theta) = (\frac{m \cdot v}{\sqrt{2\pi(av + b)}} + n) \exp(-\frac{\theta^2}{2(av + b)}) \tag{4}$$

$$m = \frac{d_{com} - d_{min}}{\frac{v_{max}}{\sqrt{2\pi(av_{max}+b)}} - \frac{v_{min}}{\sqrt{2\pi(av_{min}+b)}}}$$

$$n = d_{com} - \frac{m \cdot v_{max}}{\sqrt{2\pi(av_{max} + b)}}$$

Linear model

$$d(\theta) = m \cdot v \cdot (-a\,|\theta| + b) + n \tag{5}$$

$$m = \frac{d_{com} - d_{min}}{b \cdot (v_{max} - v_{min})}$$

$$n = d_{com} - c \cdot b \cdot v_{max}$$

For the proactively waking up frequency, the MCTA algorithm [9] uses a "refresh time" concept. Instead we adopt if the target is leaving the previous tracking subarea as the criteria: when the target gets close to the edge of the subarea, a node at the edge who detects this target will behave as the root node and broadcast an alert message asking its neighboring nodes to get prepared. When no subarea forms to cover the target, the first node who detects it will become the root node to trigger the alert. Then all the nodes in the contour with this root node as the basic point and with 4 and 5 as the scope form a subarea.

Our tracking subarea management algorithm is described as follows.

Algorithm 1: Tracking Subarea Management Algorithm

```
1 if A tracking subarea exists then
2     if The target is in the current subarea then
3         if The target is NOT leaving the subarea then
4             return;
5     Dismiss the current subarea;
6 Create a new subarea based on the $d(\theta)$ value;
7 Schedule the sleep pattern of the nodes in the new subarea following Algorithm 2;
```

Table 1. Notation of Parameters

Parameter	Definition	Default value (unit)
r	Sensor range	$10(m)$
R	Communication range	$60(m)$
v	Target speed	$[0, 30](m/s)$
TC	Waking up toggling cycle	$2(s)$
DC	Duty cycle	10%

2.4. Sleep Level Model

Usually the node's sleep level can be described by a series of sleep states $\{S_i | i \in [0, k-1]\}$, in which S_0 is the awaken state, and S_{k-1} is the deepest sleep state. The sleep state number may vary in different assumptions, e.g. [13] uses a five states sleep model, and others like [5] uses a simple two state model.

In the actual implementation, the sleep level can be embodied by the wakeup/sleep toggling cycle TC, the duty cycle DC, or the application specific definition such as [13].

In this paper we choose the two levels model – S_0 (active) and S_1 (sleep), and schedule the sleep pattern by changing the duty cycle. In Section 3 we will provide our sleep scheduling algorithm's details.

2.5. Objectives

This paper's main idea is to enhance the energy efficiency by reducing the proactively awakened sensor node quantity and scheduling these nodes' sleep pattern at the same time. Our analysis and evaluation result will show that this composite effort performs better than each single one.

3. TDSS: Target Direction-based Sleep Scheduling Algorithm

The operations of the surveillance sensor network usually include two states: the surveillance state when no target is observed, and the tracking state in response to a target's intrusion [5]. Our work is to design an energy efficient sleep scheduling algorithm for the tracking state. But at first we set up the sensor node's working mode in the surveillance state. We assume that all the nodes follow the random sleep pattern in the surveillance state. With the random sleep pattern all the nodes take the same toggling period TC and the same duty cycle DC, but each node will independently choose a random starting time of the toggle period without any collaboration. For each period, a node will wake up and keep active for $TC * DC$, and then enter the sleep state for $TC * (1 - DC)$.

For the convenience of the discussion, we assume the parameters as Table 1 and Table 2. The energy consumption data comes from the actual Mica2 platform [2, 8].

Figure 3 and Algorithm 2 describes our sleep scheduling algorithm for each single node in the tracking subarea, where X-axis is for time, and Y-axis is for the node's duty cycle. And 1) DC_{max} is the maximum duty cycle that a node will be scheduled to; 2) t_{start} is the start time of the scheduling, or the minimum time that the target moves to this node from the

Table 2. Energy consumption rate

Active (P_{sense})	Process	Send message (P_{send})
$9.6(mJ/s)$	$4(nJ/instr)$	$720(nJ/bit), 5.76(mJ/Byte)$
Receive message (P_{rcv})	Sleep (P_{sleep})	Wakeup/sleep switch (P_{switch})
$110(nJ/bit), 0.88(mJ/Byte)$	$0.33(mJ/s)$	$0.24(mJ/time)$

current position; and 3) t_{end} is the end time when the duty cycle recovers to the default value and the sleep scheduling for this node is over. 3 gives the changing trend of DC_{max}, t_{start} and t_{last} ($t_{end} - t_{start}$) when $d(\theta)$, v and $p(\theta)$ increases respectively, where $d(\theta)$, v and $p(\theta)$ are defined in Figure 1. Our sleep scheduling algorithm's details are described in Table 2.

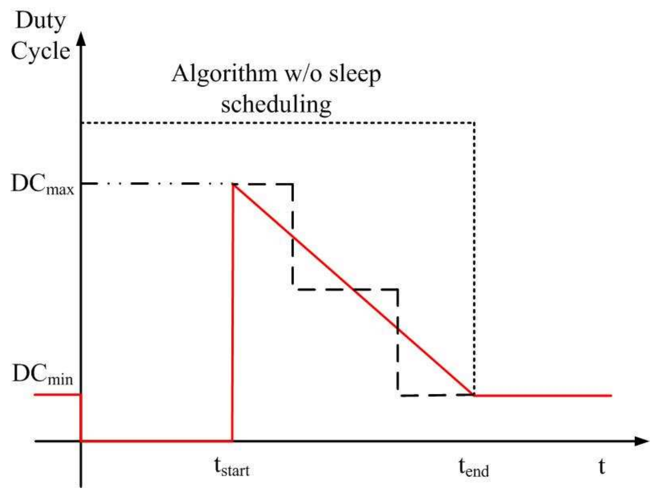

Figure 3. TDSS algorithm

The reason for each pair in Table 3 is,

- ($DC_{max} : v$) – The faster the target moves, the higher sleep level the node should keep in (more active) to avoid any missing.
- ($DC_{max} : p(\theta)$) – The higher the probability to that direction is, the more active the node should keep.
- ($t_{start} : d$) – The further the node is from the target's current position, the longer it can keep in the sleep state before the target arrives.
- ($t_{end} : v$) – The faster the target moves, the shorter the node needs to keep its high duty cycle, because the target is supposed to pass by very fast.

Next we'll give out all these parameters based on our target movement model.

Algorithm 2: TDSS – Target Direction-based Sleep Scheduling Algorithm

```
1  for each node i in the tracking subarea do
2      Compute the distance from the subarea's root node to node $i$;
3      Compute $t_{start}$ and $t_{end}$;
4      Compute the included angle $\theta$ between the "root node"-"node $i$" connection line and the instant
       velocity $v$;
5      Compute $DC_{max}$;
6      Set node $i$'s next duty cycle value as $DC_{max}$;
7      Set node $i$'s duty cycle recovery cycle number as $Round(\frac{t_{end}-t_{start}}{TC})$, where "Round" is the
       rounding function and TC is the default waking up toggling cycle;
8      if $t_{start} = minimum\_sleep\_time$ then
9          Set node $i$'s state as "SLEEP";
10         Reset node $i$'s waking up timer as $t_{start}$;
```

Table 3. Energy consumption rate

	$d(\theta)\nearrow$	$v\nearrow$	$p(\theta)\nearrow$
DC_{max}	NA	$\nearrow$	$\nearrow$
t_{start}	$\nearrow$	NA	NA
t_{last}	NA	$\searrow$	NA

3.1. DC_{max}

Since DC_{max} will increase as the target's instant speed and the direction probability increase, let $DC_{max} = i \cdot v \cdot p(\theta = 0) + j$, then DC_{max} is lineal to either v or $p(\theta)$ when the other keeps consistent.

Then for the *normal distribution* model,

$$DC_{max} = \frac{i \cdot v}{\sqrt{2\pi(av+b)}} \exp(-\frac{\theta^2}{2(av+b)}) + j \tag{6}$$

$$i = \frac{DC_{max}|_{max} - j}{v \cdot p(\theta)|_{\theta=0,v=v_{max}}} = \frac{1 - \frac{2P_{switch}}{TC\cdot(P_{sense}-P_{sleep})} - j}{\frac{v_{max}}{\sqrt{2\pi(av_{max}+b)}}}$$

$$j = DC_{max}|_{\theta=0,v=v_{max}} = DC_{min}$$

For the *linear distribution* model,

$$DC_{max} = i \cdot v \cdot (-a\,|\theta| + b) + j \tag{7}$$

$$i = \frac{DC_{max}|_{max} - j}{v \cdot p(\theta)|_{\theta=0,v=v_{max}}} = \frac{1 - \frac{2P_{switch}}{TC\cdot(P_{sense}-P_{sleep})} - j}{b \cdot v_{max}}$$

$$j = DC_{max}|_{\theta=0,v=v_{max}} = DC_{min}$$

3.2. t_{start}

t_{start} is the maximum time for the node to sleep thoroughly before the target may enter its sensing range, also it is the minimum time for the target to reach the node's sensing range with the maximum speed, thus

$$t_{start} = \frac{d-r}{v_{max}}$$

3.3. t_{end}

t_{end} is the maximum time for the node to pass by the node's sensing range, but if the target moves out of the current subarea with a shorter time than the expected passing by time, the sleep scheduling should also be recovered, thus

$$t_{end} = \min\left\{\frac{d+r}{v_{min}}, t_{out}\right\}$$

Here v_{min} is not the absolute minimum speed of the target (0), instead we define it as $v_{min} = v/2$. And t_{out} is the time point that the target moves out of the current tracking subarea.

3.4. Line AB

With DC_{max}, t_{start} and t_{end} , we can give out the line equation for AB:

$$DC = at + b \tag{8}$$

When we consider the regular case only, say, when $t_{end} = \frac{d+r}{v_{min}}$, we have

$$\begin{cases} DC_{max} = a \cdot t_{start} + b \\ DC_{min} = a \cdot t_{end} + b \end{cases}$$

Then for the *normal distribution* model,

$$\begin{cases} a = -\frac{DC_{max}-DC_{min}}{t_{end}-t_{start}} = -\frac{\frac{\sqrt{av_{max}+b}}{\sqrt{av+b}}\cdot(1-\frac{2P_{switch}}{TC\cdot(P_{sense}-P_{sleep})}-DC_{min})\cdot\frac{v}{v_{max}}\cdot\exp(-\frac{\theta^2}{2(av+b)})}{\frac{d+r}{v_{min}}-\frac{d-r}{v_{max}}} \\ b = DC_{max} - at_{start} = \frac{DC_{max}\cdot t_{end}-DC_{min}\cdot t_{start}}{t_{end}-t_{start}} \end{cases}$$

For the *linear distribution* model,

$$\begin{cases} a = -\frac{DC_{max}-DC_{min}}{t_{end}-t_{start}} = -\frac{\frac{1}{b}\cdot(1-\frac{2P_{switch}}{TC\cdot(P_{sense}-P_{sleep})}-DC_{min})\cdot\frac{v}{v_{max}}\cdot(-a|\theta|+b)}{\frac{d+r}{v_{min}}-\frac{d-r}{v_{max}}} \\ b = DC_{max} - at_{start} = \frac{DC_{max}\cdot t_{end}-DC_{min}\cdot t_{start}}{t_{end}-t_{start}} \end{cases}$$

4. Discussion

Next we discuss the analysis on the energy consumption at first, and then give out the performance analysis.

4.1. Energy Consumption Analysis

Without sleep scheduling, a node's energy will be consumed on sensing, processing, data propagating and optional communication for collaboration. But for the sleep scheduling, the extra energy is needed for proactively waking up communication e_{waking} and keeping the awakened nodes active for some time e_{active}. In our algorithm, e_{active} is the energy consumed on the increased duty cycle.

In a unit of time (i.e. one second), a single proactively waking up event's e_{waking} and e_{active} are

$$e_{waking} = \rho[(\pi R^2 - \pi r^2) \cdot P_{rcv} + \pi r^2 P_{send}] \tag{9}$$

$$\begin{aligned} e_{active} &= \textstyle\sum_{i \in A} E(\Delta DC_i \cdot t \cdot P_{sense}) \\ &= P_{sense} \textstyle\sum_{i \in A} E(\Delta DC_i \cdot t) \end{aligned} \tag{10}$$

Where ρ is the node density, A is the tracking subarea and $E()$ is the expectation. Next we calculate $E(\Delta DC_i \cdot t)$. From $DC = at + b$, thus

$$\begin{aligned} E(\Delta DC_i \cdot t) &= \textstyle\int_{t_{start}}^{t_{end}} (at + b - DC_{min}) dt \\ &= \tfrac{1}{2}(DC_{max} - DC_{min})(t_{end} - t_{start}) \end{aligned}$$

Since it's difficult to describe the node distribution in the tracking subarea, here we give an approximate result only. Let $DC_{max} = 50\%$, $d = R/2$, $v = v_{max}/2$, thus

$$\begin{aligned} \textstyle\sum_{i \in A} E(\Delta DC_i \cdot t) &= \rho S \cdot \tfrac{1}{2}(\tfrac{1}{2} - DC_{min})(\tfrac{\frac{R}{2}+r}{\frac{v_{max}}{2}} - \tfrac{\frac{R}{2}-r}{v_{max}}) \\ &= \tfrac{\rho(R+6r)(1-2DC_{min})}{8v_{max}} \cdot S \end{aligned}$$

Where S is the area of the tracking subarea.

For the *normal distribution* model,

$$S = \int_{-\pi}^{\pi} d(\theta) d\theta = \int_{-\pi}^{\pi} (\frac{m \cdot v}{\sqrt{2\pi(av+b)}} + n) \exp{(-\frac{\theta^2}{2(av+b)})} d\theta$$

For the *linear distribution* model,

$$\begin{aligned} S &= \textstyle\int_{-\pi}^{\pi} d(\theta) d\theta = \int_{-\pi}^{\pi} (m \cdot v \cdot (-a\,|\theta| + b) + n) d\theta \\ &= -amv\pi^2 + (bmv + n)\pi \end{aligned}$$

Since the normal distribution is not integrabel, we only give the detailed result for the linear distribution.

4.2. Detection Probability & Delay

In our tracking subarea algorithm, a new subarea forms when the target is going to leave the current subarea. In fact to keep tracking a target without losing, the subareas should cover all the target's moving route. Therefore, we need to guarantee there's always an intersection between two adjacent subareas. Since the subarea maximum radius is R, the communication range, we have to ensure that a target will be detected before it leaves away from the old subarea's root node for 2R.

We would like to leverage the discussion in [6] to evaluate the detection probability and detection delay. For simplicity, we'll ignore the slow target case for which the system can absolutely provide a higher detection probability and a lower detection delay, that's to say, the discussion below for the fast target case provides a upper bound for all the target speed distribution.

In [6], the probability that a single node can detect a target within L distance is

$$P = DC + \frac{\pi r L}{(2L + \pi r/2) \cdot v \cdot TC}$$

Since we have to guarantee to detect the target before it leaves away from the old cluster header for $2R$, here let $L = R$. Thus

$$P = DC + \frac{\pi r R}{(2R + \pi r/2) \cdot v \cdot TC}$$

However, this is the case without sleep scheduling where the node's toggle cycle and duty cycle will not change at all. For our TDSS algorithm, we need to normalize P in the tracking subarea.

$$P_{TDSS} = \frac{1}{d(\theta)} \int_0^{d(\theta)} \frac{DC_{max} - DC_{min}}{2} dx + \frac{\pi r R}{(2R + \pi r/2) \cdot v \cdot TC}$$

5. Performance Evaluation

5.1. Simulation Environment

Our simulation environment is as follows.

- Network scale – 2,000 nodes are deployed in a grid structure of 20 rows and 100 columns. The distance between two adjacent nodes is 5 (m).
- Target motion – Two model options to describe the actual target motion: uniform rectilinear motion (URM) and uniform curvilinear motion (UCM).
- Target speed – 10 target speeds in total which is an arithmetic progression $\{3, 6, \ldots, 30\}$.
- Algorithms – We simulate four algorithms: legacy circle-based scheme, MCTA, TDSS (normal distribution) and TDSS (linear distribution).

Thus we simulate 80 profiles in total ($2 targetmodels \times 10 targetspeeds \times 4 algorithms$). Other parameters follow Table 1 and Table 2.

Here the target motion model is totally different from what we used in Section 2.2. Section 2.2 presents the target movement model that we will use in TDSS algorithm to make our algorithm match the actual object motion as possible. However, we cannot use the same model when simulating the target motion, otherwise everything would be perfect. Instead we use physics rules to describe the target motion.

Let the target move with a random turning angle with the interval $[0, \alpha_{max}]$, where α_{max} decreases as the speed increases. We use a linear mapping to describe their relation where $\alpha = 25^o$ when $v = 9(m/s)$, and $\alpha = 5^o$ when $v = 30(m/s)$. Then use a time granularity in a lower order of magnitude than the sensor nodes' waking up toggling cycle to describe the target motion. This will guarantee the precision of our simulation.

Figure 4 shows an example of the target moving route with the speed $15(m/s)$ in a local area of $100m \times 100m$.

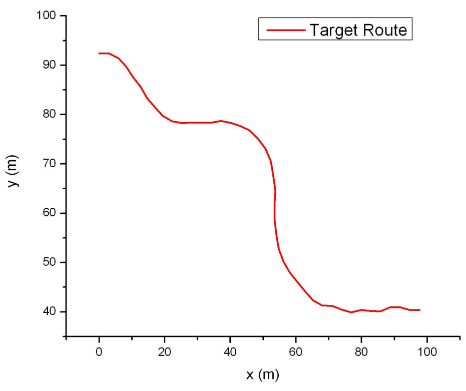

Figure 4. Target Moving Route

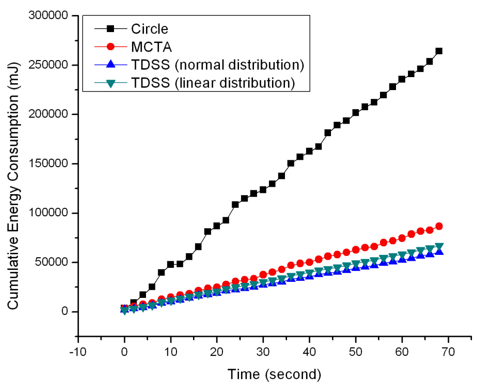

Figure 5. Cumulative Energy Consumption

5.2. Simulation Output

Since our major objective is to enhance the energy efficiency by combining the working nodes reducting efforts and the sleep scheduling algorithm, the key contribution of the TDSS algorithm is the energy consumption reduction.

First Figure 5 presents the energy consumption as time goes on. The figure clearly shows that TDSS algorithm saves more energy than other algorithms.

Also we use a concept of Effective Energy Consumption Percentage (EECP) to describe this energy efficiency. The background behind this is: usually the proactively waking up mechanism will awaken the neighboring nodes to get prepared for the approaching target, however not all the awakened nodes can detect the target, then its energy consumed on keeping active will be wasted. EECP is the percentage of the energy consumed by those nodes who detect the target actually among all the energy consumed to keep all the neighboring nodes active. The relationship between EECP and the target speed under all the two target motion models and all the four algorithms are shown in Figure 6 and Figure 7.

From Figure 6 and Figure 7, we can find that TDSS exceeds both the legacy circle-based scheme and MCTA on the energy efficiency by $(10\% - 200\%)$. Table 4 lists the advantage of TDSS (normal distribution) under several target speeds and the URM motion model as

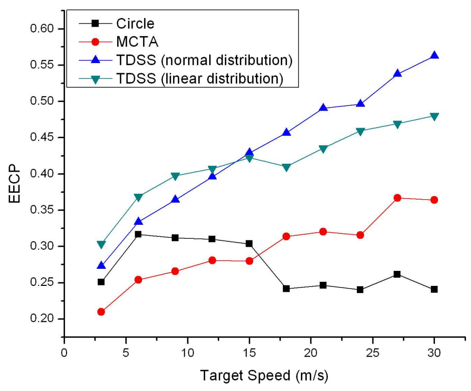

Figure 6. EECP vs. Target Speed (URM)

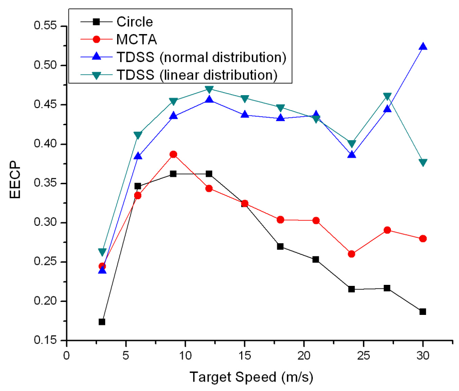

Figure 7. EECP vs. Target Speed (UCM)

Table 4. The advantage of TDSS (normal distribution) compared with other algorithms

Other algorithm	$3(m/s)$	$15(m/s)$	$30(m/s)$
Circle	8.85%	41.45%	133.92%
MCTA	29.97%	53.27%	54.51%

an example. This is the most important contribution of this paper. Also from Figure 6 and Figure 7, we can tell that when the target moves in a low speed ($roughly \leq 15(m/s)$), TDSS with the linear distribution behaves better than the normal distribution model. On the contrary TDSS with the normal distribution model behaves better with a high speed ($roughly > 15(m/s)$).

Sure with the sleep scheduling, there may be some performance loss. Figure 8 gives out the detection delay in the case of URM model and the target speed $30(m/s)$. It shows that TDSS (normal distribution) has some detection timeliness loss, but TDSS (linear distribution) performs similar as the legacy algorithm and MCTA.

Also we define a new concept “Tracking Degree” (TD) to describe the detection probability, since the overall detection probability is hard to describe directly. TD equals to the covered (by sensor nodes) route length divided by the target's total route length. The bigger TD is, the higher the detection probability would be. Figure 9 shows the relationship between TD and the energy consumption (EC) in the case of UCM model and target speed $15(m/s)$. We may find that in terms of the detection probability, all the algorithms are very similar. From the discussion above, we can tell that TDSS achieves much better energy efficiency with little performance loss.

6. Conclusion & Future Work

This paper presents a target direction-based sleep scheduling algorithm (TDSS) for a mobile target tracking surveillance sensor network. We achieve the energy efficiency by combin-

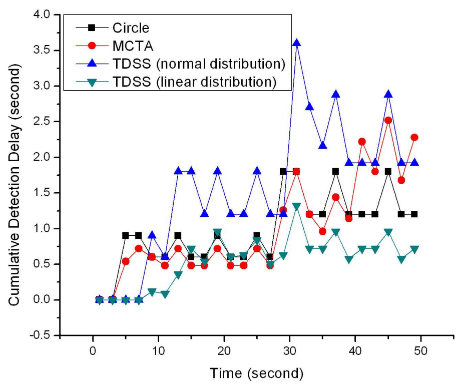

Figure 8. Cumulative Detection Delay

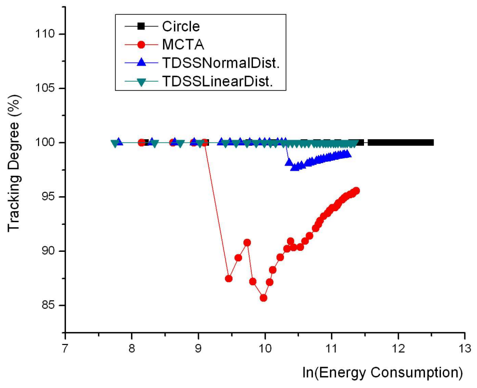

Figure 9. Tracking Degree vs. EC

ing the working node reducing efforts and the sleep scheduling. For the sleep scheduling, we take the target's moving direction into account by introducing two target moving direction probability distributions: normal distribution and linear distribution. This direction-based scheduling algorithm can make the tracking subarea be more suitable to describe the actual object's motion. Finally we compare TDSS's two distribution models with the legacy circle-based proactively waking up scheme and MCTA. The evaluation result shows that TDSS can achieve much better energy efficiency but with little loss on the detection probability and the detection delay.

Our future work would include but not limited to:

- Balance of the tracking performance and the network lifetime – For example, given a detection probability's lower bound or a detection delay's upper bound, how to prolong the network lifetime?
- Object motion model – How to simulate a target's motion more likely to reduce the unnecessary energy consumption?
- Node collaboration – During the tracking process, how can the subsequent nodes leverage the already available target information propagated by the anterior nodes? The information accumulation may dismiss the reduplicate workload thus achieve a better balance.

References

[1] 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, 2004.

[2] M. Athanassoulis, I. Alagiannis, and S. Hadjiefthymiades. Energy efficiency in wireless sensor networks: A utility-based architecture. In *European Wireless 2007*, 2007.

[3] 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*, number 4, 2005.

[4] J. Fuemmeler and V. Veeravalli. Smart sleeping policies for energy efficient tracking in sensor networks. *IEEE Transactions on Signal Processing*, 2007.

[5] 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*, pages 129–143, 2004.

[6] T. He, P. Vicaire, T. Yan, and L. L. et al. Achieving real-time target tracking using wireless sensor networks. *ACM Transaction on Embedded Computing System (TECS)*, 2007.

[7] J. Hightower and G. Borriello. Location systems for ubiquitous computing. *IEEE Computer*, 34(8):57–66, August 2001.

[8] J. L. Hill and D. E. Culler. Mica: a wireless platform for deeply embedded networks. *IEEE Micro*, 22, 2002.

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

[10] R. Stoleru, J. A. Stankovic, and S. Son. Robust node localization for wireless sensor networks. In *EmNets*, 2007.

[11] D. Tian and N. Georganas. A coverage-preserving node scheduling scheme for large wireless sensor networks, 2002.

[12] 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*, 7:1193–1215, 2007.

[13] X. Wang, J.-J. Ma, S. Wang, and D.-W. Bi. Prediction-based dynamic energy management in wireless sensor networks, 2007.