

ACOS: An Area-based Collaborative Sleeping Protocol for Wireless Sensor Networks

Yanli Cai¹, Minglu Li¹, Wei Shu², and Min-You Wu^{1,2}

¹Department of Computer Science and Engineering
Shanghai Jiao Tong University, Shanghai 200030, China

²Department of Electrical and Computer Engineering
The University of New Mexico, Albuquerque, New Mexico, USA

{cai-yanli, li-ml, wu-my}@cs.sjtu.edu.cn, shu@ece.unm.edu

Abstract: A surveillance application requires sufficient coverage of the protected region while minimizing the energy consumption and extending the lifetime of sensor networks. This can be achieved by putting redundant sensor nodes to sleep. In this paper, we propose a precise and energy-aware coverage control protocol, named Area-based Collaborative Sleeping (ACOS). The ACOS protocol, based on the net sensing area of a sensor, controls the mode of sensors to maximize the coverage, minimize the energy consumption, and to extend the lifetime of the sensor network. The simulation shows that our protocol has better coverage of the surveillance area while waking fewer sensors than other state-of-the-art sleeping protocols. It extends the lifetime of sensor networks significantly.

Keywords: Sensor Networks, Coverage, Energy Conservation, Lifetime

1. Introduction

A wireless sensor network consists of a set of inexpensive sensors with wireless networking capability [1]. Applications of wireless sensor networks include battlefield surveillance, environment monitoring and so on [2]. Recently, a lot of research activities have been dedicated to wireless sensor networks.

As sensors may be distributed arbitrarily, one of the fundamental issues in wireless sensor networks is the coverage problem. The coverage of a sensor network, measured by the fraction of the region covered, represents how well a region of interest is monitored. The degree of coverage needed depends on specific applications. In applications such as military surveillance, it is necessary to provide as much coverage to a security-sensitive region as possible.

On the other hand, a typical sensor node such as an individual mote, can only last 100-120 hours on a pair of AA batteries in the active mode [3]. Power sources of the sensor nodes are non-rechargeable in most cases. However, a sensor network is usually desired to last for months or years. Thus, power conservation is a challenging problem in wireless sensor networks. A natural and feasible way is to put redundant sensors to sleep while keeping a

certain degree of coverage. Sleeping protocols to save energy are under intensive study, such as RIS [4, 5], PEAS [6] and PECAS [4]. These protocols presented different approaches to utilizing resources, but needs further improvement in coverage or efficient energy consumption.

Here we propose a sleeping protocol, named Area-based Collaborative Sleeping or ACOS. This protocol precisely controls the mode of sensors to maximize the coverage and minimize the energy consumption based on the *net sensing area* of a sensor. The net sensing area of a sensor is the area of the region exclusively covered by the sensor itself. If the net sensing area of a sensor is less than a given threshold, the sensor will go to sleep. Collaboration is introduced to the protocol to balance the energy consumption among sensors. Performance study shows that ACOS has better coverage of the surveillance area while waking fewer sensors than other state-of-the-art sleeping protocols.

The rest of the paper is organized as follows. Section 2 discusses previous research related to the coverage problem. Section 3 describes the basic design of our protocol. Section 4 improves the baseline ACOS for better performance. Section 5 provides a detailed performance evaluation and comparison with other state-of-the-art protocols. We conclude the paper in Section 6.

2. Related Work

Different coverage methods and models have been surveyed in [7, 8, 9]. Three coverage measures are defined [7], which are area coverage, node coverage, and detectability. Area coverage represents the fraction of the region covered by sensors and node coverage represents the number of sensors that can be removed without reducing the covered area, while detectability shows the capability of the sensor network to detect objects moving in the network. Centralized algorithms to find exposure paths within the covered field are presented in [8]. In [9], the authors investigate the problem of how well a target can be monitored over a time period while it moves along an arbitrary path with an arbitrary velocity in a sensor network. A given belt region is said to be k -barrier covered with a sensor network if all crossing paths through the region are k -covered, where a crossing path is any path that crosses the width of the region completely [10].

Power conservation protocols such as GAF [11], SPAN [12] and ASCENT [13] have been proposed for ad hoc multi-hop wireless networks. They aim at reducing the unnecessary energy consumption during the packet delivery process. In [14], a heuristic is proposed to select mutually exclusive sets of sensors such that each set of sensors can provide a complete coverage. In [15], redundant sensors that are fully covered by other sensors are turned off to reduce power consumption, while the fraction of the area covered by sensors is preserved. The problem of providing periodic energy-efficient radio sleep cycles while minimizing the end-to-end communication delays is addressed in [16]. Target coverage problem has been studied in sensor surveillance networks where a set of sensors and targets are deployed in [17, 18]. In [17], heuristics are proposed to divide the sensor nodes into a number of sets, which are activated successively, and at any time instant only one set is active. In [18], target watching timetable for each sensor is built to achieve maximal lifetime.

Sleeping protocols such as RIS [4, 5], PEAS [6] and PECAS [4] have been proposed to extend the lifetime of sensor networks. In RIS, each sensor independently follows its own sleep schedule which is set up during network initialization. In PEAS, a sensor sends a probe message within a certain probing range when it wakes up. The active sensor replies to any received probe message. The sensor goes back to sleep if it receives replies to its probes. In PEAS, an active node remains awake continuously until it dies. PECAS makes an extension to PEAS. Every sensor remains within active mode only for a duration and then goes to sleep.

3. Basic Protocol Design

In this section, we describe the basic design of the Area-based Collaborative Sleeping (ACOS) protocol. This protocol precisely controls the mode of sensors so that the coverage of the sensor network can be maximized and the energy consumption minimized, therefore, the lifetime of the sensor network extended.

3.1 Notations and Assumptions

We adopt the following notations and assumptions throughout the paper.

- Consider a set of sensors $S = \{s_1, s_2, \dots, s_n\}$, distributed in a two-dimensional Euclidean plane.
- Sensor s_j is referred as a *neighbor* of another sensor s_i , or vice versa, if the Euclidean distance between s_i and s_j is less than $2r$.
- Assume that each sensor knows its own location [19, 20, 21, 22]. As shown in Section 3.2, relative locations [23] are used for our protocol.
- A sensor has two power consuming modes: low-power mode and active mode.
- For simplicity, the *sensing region* of each sensor is a disk centered at the sensor with radius r , called the *sensing range*. As shown later, the protocol can easily extend to an irregular sensing region.
- The *net sensing region* of sensor s_i is the region in the sensing range of s_i but not in the sensing range of any other active sensor. The *net sensing area* or *net area* of s_i is the area of the net sensing region. The *net area ratio*, denoted as β_i , is the ratio of s_i 's net sensing area to s_i 's maximal sensing area, πr^2 .
- The *net area Threshold*, denoted as φ , is a parameter between 0 and 1.

3.2 The Net Area Calculation

The ACOS protocol is based on precise calculation of the net sensing area of a sensor. The shadowed region with bold boundary in Fig.1 shows an example of the net sensing region of sensor s_θ . Before discussion of ACOS, a solution to computing the net area is presented here. The problem to compute the net area of sensor s_θ can be solved by two steps. In the first step, the boundaries of each net sensing region of s_θ are determined, and in the second step, the area of each net sensing region is calculated.

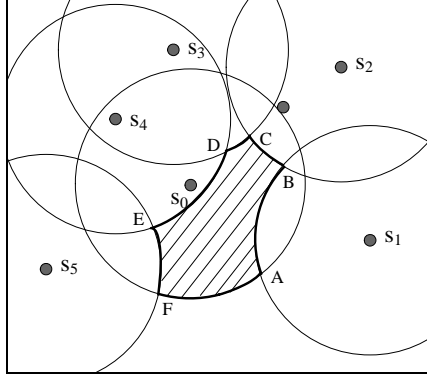


Fig.1. The net sensing region of a sensor s_0

In the first step, the algorithm in [24] is used to find boundaries of the net sensing regions inside a sensor, which is referred as *perimeter-coverage algorithm*. The perimeter-coverage algorithm costs polynomial time to find the coverage of the protected region by considering how the perimeter of each sensor's sensing region is covered. Consider sensor s_i , a segment of s_i 's perimeter is *k-perimeter-covered* if all points on the segment are in the sensing range of at least k sensors other than s_i itself. This algorithm can be used to find how many segments of s_i 's perimeter are divided by its neighbors and whether each of these segments is *k-perimeter-covered* or not.

Consider s_0 as shown in Fig.1, we first find the 0-perimeter-covered segments of s_0 's perimeter. In this example, there is only one of these segments, the minor arc \widehat{FA} . Then in the sensing range of s_0 , we find the 1-perimeter-covered segments for each of s_0 's neighbors, the minor arcs \widehat{AB} , \widehat{BC} , \widehat{CD} , \widehat{DE} , \widehat{EF} in this example. After all the segments are found, two segments are jointed together if they have common end points. The closed boundary of each net sensing region is determined by a segment sequence.

In the second step, the area of each net sensing region can be computed by calculating the area of the polygon formed by its segment sequence, the polygon $ABCDEF$ in this example, and the area of the region between each segment of arc and the corresponding chord. Here, each node only needs to know the relative locations of its neighbors. To determine the relative location among sensors is easier than determining each node's absolute location.

3.3 Basic Protocol Design

The basic design of ACOS protocol is presented in this section, which is called *baseline ACOS*, leaving other issues to be addressed in Section 4.

The lifetime of a sensor network is affected not only by the total energy consumption of all the sensors but also by the balance of energy consumption among sensor nodes. The earlier some nodes run out of energy than others, the more quickly the density of functional sensors drops, and the shorter the network can last upon certain coverage. Thus, collaboration strategies should be introduced into sleeping protocols to balance energy consumption among neighboring nodes. The strategy is to put the sensor to sleep after it has worked for a given time $T_{wake_Duration}$. The neighboring nodes wake up later and take the turn to serve in the active mode. This strategy is used in PECAS, but there may exist *sensing holes* before the

neighboring nodes take its turn, since a sensor wakes at most $T_{wake_Duration}$ time. Our protocol takes a different strategy to prevent this problem. After working for $T_{wake_Duration}$, the sensor remains in the active mode until its sleeping neighbors wake up and replace it. The net area threshold φ varies from 0 to 1, and is set by applications depending on the degree of coverage required. An application requiring higher degree of coverage may choose a smaller φ , thus more sensors will be in active mode.

Each sensor node has four states: Sleep state, PreWakeUp state, Awake state, and Overdue state. The Sleep state corresponds to the low-power mode. The PreWakeUp, Awake, and Overdue states belong to the active mode. The PreWakeUp is a transit state and lasts for a short period of time, while the Awake and Overdue states may last for several minutes or hours. Every sensor remains in the Awake state for no more than $T_{wake_Duration}$. The state transition diagram of the baseline ACOS is show in Fig.2.

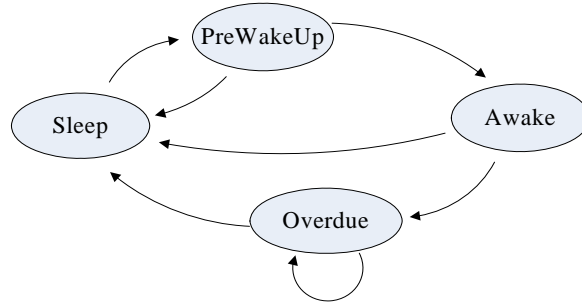


Fig.2. State transition diagram of basic ACOS

Sensor s_i has a decreasing sleep timer $T_{sleep_left}^i$, which represents the time left before s_i wakes up again and a decreasing wake timer $T_{wake_left}^i$, which is initialized as $T_{wake_Duration}$ when the sensor turns from the low-power mode to the active mode. The value of $T_{wake_Duration} - T_{wake_left}^i$ indicates how long the sensor has been in the active mode since it turned from the low-power mode to the active mode. Sensor s_i maintains an active neighbor list $nList^i$ by *collecting* information from every message received. For any neighbor s_k in $nList^i$, s_k 's location and $T_{wake_left}^k$ are also stored in $nList^i$. All the above timers decrease as the time progresses, as shown in event $e0$ in Fig.3.

For any sensor s_i

Event e0: the clock of s_i ticks one time

```

if( $s_i$  is in Sleep state){
     $s_i$ 's sleep timer  $T_{sleep\_left}^i = T_{sleep\_left}^i - 1$ ;
} else{
     $s_i$ 's wake timer  $T_{wake\_left}^i = T_{wake\_left}^i - 1$ ;
    Update the timers in local neighbor list, for any  $s_k \in nList^i$ ,  $T_{wake\_left}^k = T_{wake\_left}^k - 1$ ;
}

```

When sensor s_i is in Sleep state

Event e1: the sensor s_i 's sleep timer $T_{sleep_left}^i$ has decreased to zero

```

Change to PreWakeUp state;
Broadcast a PreWakeUp_Msg within radius  $2r$ ;
Within  $T_w$  seconds, upon receipt of a Reply_PreWakeUp_Msg from neighbor  $s_j$ ,
extract the location of  $s_j$  and  $T_{wake\_left}^j$  and store them into  $nList^i$ ;
Compute the net area ratio  $\beta_i$ ;
if ( $\beta_i < \varphi$ ){
    Set  $T_{sleep\_left}^i = \mathbf{Min}\{ T_{wake\_left}^k, \text{ for } s_k \in nList^i \} + \varepsilon$ ,  $\varepsilon$  is a random offset;
    Clear  $nList^i$  and change back to Sleep state;
} else{
    Change to Awake state;
    Set timer  $T_{wake\_left}^i = T_{Wake\_Duration}$ ;
    Broadcast a Wake_Notification_Msg including its location and  $T_{wake\_left}^i$  within
radius  $2r$ ;
}

```

When sensor s_i is in Awake state

Event e2: sensor s_i receives a PreWakeUp_Msg from s_j

```

Reply  $s_j$  with Reply_PreWakeUp_Msg, including its location and  $T_{wake\_left}^i$ ;

```

Event e3: $T_{wake_left}^i$ of the sensor s_i has decreased to zero

```

Change to Overdue state;

```

When sensor s_i is in Awake or Overdue state

Event e4: sensor s_i receives a Sleep_Notification_Msg from s_j

```

Remove  $s_j$  from  $nList^i$ ;

```

Event e5: sensor s_i receives a Wake_Notification_Msg

```

Update  $nList^i$ ;
Compute the net area ratio  $\beta_i$ ;
if ( $\beta_i < \varphi$ ){
    Broadcast a Sleep_Notification_Msg within radius  $2r$ ;
    Set  $T_{sleep\_left}^i = \mathbf{Min}\{ T_{wake\_left}^k, \text{ for } s_k \in nList^i \} + \varepsilon$ ,  $\varepsilon$  is a random offset;
    Clear  $nList^i$  and change to Sleep state;
}

```

Fig.3. The events of basic ACOS

When s_i wakes up, its state changes from Sleep to PreWakeUp. It broadcasts a message PreWakeUp_Msg to its neighbors within radius $2r$ and waits for T_w seconds. When any neighboring sensor s_j is in Awake state and receives this message, s_j sends back a Reply_PreWakeUp_Msg including its location and $T_{wake_left}^i$ into the message. Upon receipt of a Reply_PreWakeUp_Msg from any neighbor s_j , s_i extracts the location of s_j and $T_{wake_left}^j$ and stores them into its $nList^i$. At the end of T_w , s_i computes the net area ratio a_i .

If β_i is less than φ , it shows that s_i is not contributing enough coverage, and it is unnecessary for s_i to work at this moment. So it returns back to Sleep state and sleeps for a period time of the minimum value of all $T_{wake_left}^j$, from $nList^i$. It is possible that several neighbors around an active sensor s_j get its $T_{wake_left}^j$ and all wake up at the same time. The consequence is that not only they contend with communications, but also most of them may decide to start to work because of unawareness of each other. To avoid this situation, a random offset ε can be added to the sleep time.

If β_i is equal or greater than φ , s_i changes to Awake state, initialize its wake timer $T_{wake_left}^i$ and broadcasts a Wake_Notification_Msg including its location and $T_{wake_left}^i$ to its neighbors, as described by event $e1$ in Fig.3.

When s_i is still in the Awake state and hears a PreWakeUp_Msg from s_j , it replies s_j with a Reply_PreWakeUp_Msg, including its $T_{wake_left}^i$. Although sensor s_i in its Overdue state is also in the active mode, it does not reply to PreWakeUp_Msg so that s_i is not counted by newly waked-up sensors and is more likely able to go to sleep in a short time. This is how energy consumption balance among sensors is achieved. This procedure is described in event $e2$ in Fig.3.

When s_i is in the Awake state and its wake timer $T_{wake_left}^i$ has decreased to zero, it changes from Awake to Overdue state, as shown in event $e3$ in Fig.3.

When s_i is in the Awake or Overdue state and hears a Wake_Notification_Msg, it updates its list $nList^i$ and recalculates the net area ratio β_i first. If β_i is less than φ , this indicates that s_i can go to sleep safely, and therefore broadcasts a Sleep_Notification_Msg to its neighbors. Then it changes to the Sleep state and sleep for the minimum value of all $T_{wake_left}^k$, from $nList^i$. Also, a random offset is added to the sleep time. This procedure is described in event $e5$ in Fig.3.

If s_i is in the Awake or Overdue state and hears a Sleep_Notification_Msg from s_j , s_i removes s_j entry from $nList^i$, as shown in event $e4$ in Fig.3.

Fig.3 demonstrates all events that occur in our protocol. The events drive a sensor to change from one state to another and precisely control its power consuming modes.

However, two problems are not taken into consideration in the basic design of the ACOS protocol initially. One is the unawareness of dead neighbors when a sensor decides to go to sleep, and the other is about sleep competition caused by waking up of a sensor. We deal with these two problems in Section 4.

3.4 Extension to Irregular Sensing Regions

The sensing region depends on the geographical location and could be irregular. Our protocol can be easily extended to irregular sensing regions under the condition that each sensor's sensing region is known. We can still use our net area calculation method to find out the boundaries of net sensing regions for each sensor. Then the area of the net sensing region can be computed by using polygon approximation.

4. Optimizations

As we have mentioned in Section 3.3, there are two issues to be addressed. The first is the unawareness of dead neighbors. When a sensor s_i receives a Wake_Notification_Msg, it computes its net area ratio β_i . The calculation of β_i depends on information stored in the local neighbor list $nList^i$. The information may be outdated, because some of neighbors may have died from physical failure or energy depletion without notification. The second problem is about sleep competition caused by a waking up sensor. Consider sensor s_i which decides to wake up after computing the net sensing area ratio β_i . It then broadcasts a Wake_Notification_Msg to its neighbors, and several neighbors may receive this message. Each of them computes their net area ratio without collaboration, and many of them may go to sleep. We call this situation *multiple sleeps*. In some cases, multiple sleeps are needed to reduce overlap, but in other cases, multiple sleeps should be avoided.

In this Section, we modify the baseline ACOS to solve the problem of dead neighbors and to reduce the effect of multiple-sleeps problem by adding a new transit PreSleep state. When a sensor s_i receives a Wake_Notification_Msg and its net area ratio β_i is less than φ , it does not turn to sleep immediately but changes to a PreSleep state. In PreSleep state, s_i recollects its neighbors' information and decides whether it goes to sleep or turns back to the former state. The state transition diagram for enhanced ACOS is shown in Fig.4.

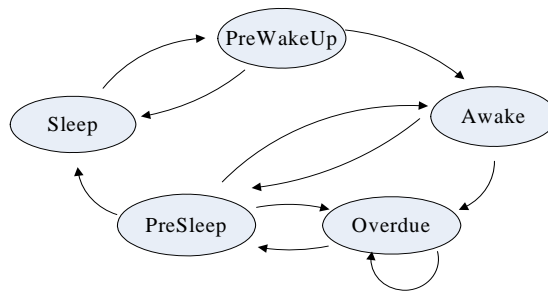


Fig.4. State transition diagram for optimized ACOS

4.1 Dealing with Dead Neighbors

Consider a sensor s_i in its active mode, if some of its neighbors die and result in some new uncovered region, what s_i can help is to continue to work as needed, even longer than the previously planned wake-up time.

When s_i receives a Wake_Notification_Msg and its net area ratio β_i is less than φ , it changes to PreSleep state and clears the current information in its $nList^i$. Then it broadcasts a

PreSleep_Msg to its neighbors and waits for T_w seconds. When a neighbor s_j , in its Awake or Overdue state, hears this message, s_j sends back a Reply_PreSleep_Msg including its location and $T_{wake_left}^i$. At the end of T_w , s_i recomputes the net area ratio β_i' . If β_i' is equal or greater than φ , this indicates that some neighbors died after the last time s_i woke up and that s_i should not go to sleep at the moment. This procedure is shown in the early part of event e5 and in event e6 in Fig.5.

The followings are modified events to the baseline ACOS

Event e5: sensor s_i in Awake or Overdue state receives a Wake_Notification_Msg from s_j

 Update the local neighbor list $nList^i$;

 Compute the net area ratio β_i ;

if ($\beta_i < \varphi$) {

 Change to PreSleep state and clear $nList^i$, broadcast a PreSleep_Msg within radius $2r$;

 Within T_w seconds, receive Reply_PreSleep_Msg and update $nList^i$;

 Recompute the net area ratio β_i' using updated $nList^i$;

if ($\beta_i' < \varphi$) {

 Broadcast a SleepIntent_Msg within radius $2r$, including β_i' ;

 Within T_w seconds, receive SleepIntent_Msg from others;

 Choose s_k which has the minimum net area ratio β_k' among all the SleepIntent_Msg;

if ($\beta_i' \geq \beta_k'$) Recompute the net area ratio β_i'' without s_k ; else $\beta_i'' = 1$

if ($\beta_i' < \beta_k'$ or $\beta_i'' < \varphi$) {

 Broadcast a Sleep_Notification_Msg within radius $2r$;

 Set $T_{sleep_left}^i = \mathbf{Min}\{ T_{wake_left}^l, \text{ for } s_l \in nList^i \} + \varepsilon$, ε is a random offset;

 Clear $nList^i$ and change to Sleep state

 } **else** {

 Change back to the former state;

 }

else {

 Change back to the former state;

 }

 }

 }

Event e6: sensor s_i in Awake or Overdue state receives a PreSleep_Msg from s_j

 Reply s_j with Reply_PreSleep_Msg, including its location and $T_{wake_left}^i$;

Fig.5. The modified events to basic ACOS

4.2 Dealing with Multiple Sleeps Caused by a Waking Up Sensor

The multiple-sleeps problem is harder than the dead-neighbor problem. In some cases, multiple sleeps are necessary to reduce overlap, while in other cases multiple sleeps will cause sensing holes and should be avoided. Fig.6 shows examples of these two cases.

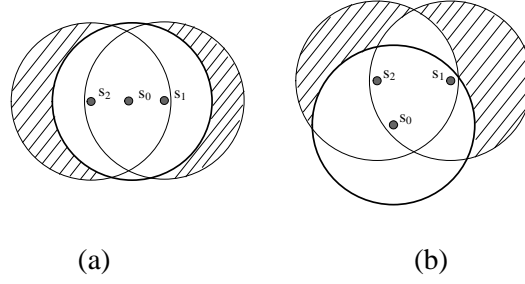


Fig.6. Example of two cases in multiple sleeps problem

In Fig.6, s_0 is a sensor that just woke up and broadcasted a `Wake_Notification_Msg`. The shadowed regions are net sensing regions of sensors s_1 and s_2 in their Overdue state. We set $\varphi=0.5$ in this example. Because s_1 and s_2 are not counted when their neighbor s_0 woke up, s_0 decides to turn to work. According to baseline ACOS, both s_1 and s_2 in Fig.6(a) and Fig.6(b) should go to sleep, as the net area ratios of s_1 and s_2 are less than φ , respectively. From Fig.6(a), we can see that both s_1 and s_2 could go to sleep, because the sleep of s_1 does not increase s_2 's net area too much or any at all. However, in Fig.6(b), s_2 should go to sleep as it has less net area than s_1 . After the sleep of s_2 in Fig.6(b), s_1 ' net area ratio β_1 increases and becomes greater than φ . Thus, s_1 should not go to sleep.

The protocol is enhanced by making the neighbors that are ready to sleep collaborate with each other. When sensor s_i receives a `Wake_Notification_Msg` from s_j , it updates its net area ratio β_i' and broadcasts a `SleepIntent_Msg` to its neighbors, including β_i' . Within T_w seconds, it receives `SleepIntent_Msg` from its neighbors, who are intent to sleep too. At the end of T_w seconds, it chooses the sensor s_k who has the minimum value of β among the neighbors from whom a `SleepIntent_Msg` had been received.

If β' is greater than s_k 's net area ratio β_k' , then s_i does not hold the minimum net area ratio and its neighbor s_k may go to sleep. Then s_i recomputes the net area ratio β_i'' by regarding s_k as a sleep node. If β_i'' is less than β_k' , it indicates that s_i does have the minimum net area ratio. If β_i'' is less than φ , it indicates the sleep of s_k does not largely increase s_i 's net area. So s_i could go to sleep relatively safely in the case of $\beta_i'' < \beta_k'$ or $\beta_i'' < \varphi$. This procedure is also shown in the last part of event $e5$ in Fig.5. A more efficient strategy to select a set of s_j 's neighbors and enable them to sleep is currently studied, in order to achieve better coverage while making more neighbors of sleep.

5. Performance Evaluation

In this section, we evaluate ACOS protocol and compare it with other three protocols RIS, PEAS and PECAS. We present the power consumption model for simulation in Section 5.1. How to choose $T_{wake_Duration}$ and net area threshold in ACOS protocol is discussed in Section 5.2. We compare the coverage and the network lifetime with other protocols in Section 5.3.

In our simulation, the sensing range of each sensor is set as 20 meters, i.e. $r = 20m$, and the communication range is $40m$. The sensors are uniformly distributed in a $400m \times 400m$

region, with bottom-left coordinate (0, 0) and top-right coordinate (400, 400). In order to evaluate the relations between coverage and different node densities, the numbers of distributed sensors are 400, 800, 1600 and 3200, respectively with density 1, 2, 4 and 8 per square of $r \times r$. From now on, we will abbreviate “square of $r \times r$ ” as “r-square.”

5.1 Energy Consumption Model

For a sensor node, the energy consumption mainly consists of three parts: the processor, radio and sensors such as the sounder, microphone. The processor typically has two power levels when it is in active and sleep mode. The radio at least has four power levels corresponding to the following states: transmitting, receiving, idle listening and sleeping. Typically, the power required to idle listening is about the same as receiving. Each sensor on a node usually consumes quite a little energy comparing to the processor or radio. The sleeping power of a sensor node component is usually two to four orders of magnitude less than the active power.

According to Mica2 Mote sensor nodes [3], we set up the energy consumption levels of different components, as shown in Table 1. Most components work between 2.2V to 3.3V. For simplicity, we assume each component works at 2.5V. As several kinds of sensors may work together to finish a task, such as sensor positioning by using sounder and microphone, we assume that the current of all active sensors on one node is 5mA.

Table 1. Energy Consumption Levels

| Component | Current | Power |
|----------------------|---------|---------|
| Processor active | 8 mA | 20 mW |
| Processor sleep | 15 uA | 37.5 uW |
| Radio transmit | 27 mA | 67.5 mW |
| Radio receive | 10 mA | 25 mW |
| Radio sleep | 1 uA | 2.5 uW |
| Radio idle listening | 10 mA | 25 mW |
| Sensors active | 5 mA | 12.5 mW |
| Sensors sleep | 5 uA | 12.5 uW |

We also assume the battery of a sensor node can last for 100 hours when the node keeps the processor, radio and sensors all active all the time. We assume the data transmission rate is 19.2kbps.

5.2 Parameters Tuning of ACOS protocol

5.2.1 Wake Duration

The selection of wake duration $T_{wake_Duration}$ is important for the ACOS protocol. If $T_{wake_Duration}$ is too long, the protocol cannot balance the energy consumption among sensor nodes. Shorter $T_{wake_Duration}$ will result in larger overhead. The main overhead comprises radio

idle listening, communication and net area computing. Radio idle listening and communication in the PreWakeUp state cost most overhead. Communication in other states, net area computing and other computing in the protocol take up the rest overhead.

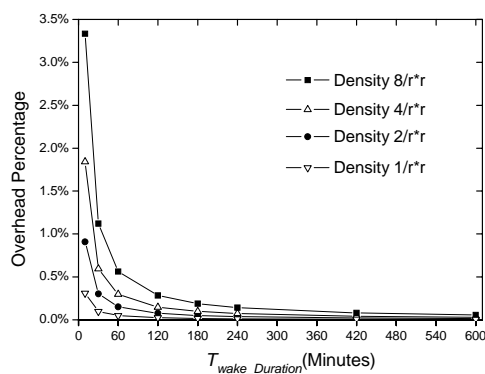


Fig.7. The protocol overhead with regard to wake duration

Fig.7. shows the relationship of $T_{wake_Duration}$ and the overhead percentage, the ratio of overhead to all energy consumption. Net area threshold φ is set to 0.1, however our simulation shows that the overhead percentage of different net area thresholds is almost the same. As shown in Fig.7, the overhead percentage is proportional to the node density, thus this protocol may not suitable when the node density is particularly high. With the same node density, the overhead percentage decreases sharply as the $T_{wake_Duration}$ increases. When $T_{wake_Duration} = 240$ minutes, the overhead percentage decreases to 0.14% with node density of 8 per r-square, while it decreases to 0.038% with node density of 2 per r-square. This result shows that as long as the wake duration is properly tuned, the overhead is neglectable.

5.2.2 Net Area Threshold

For each density, we take samples when the network runs stably after the initialization stage and before any sensor node depletes its energy.

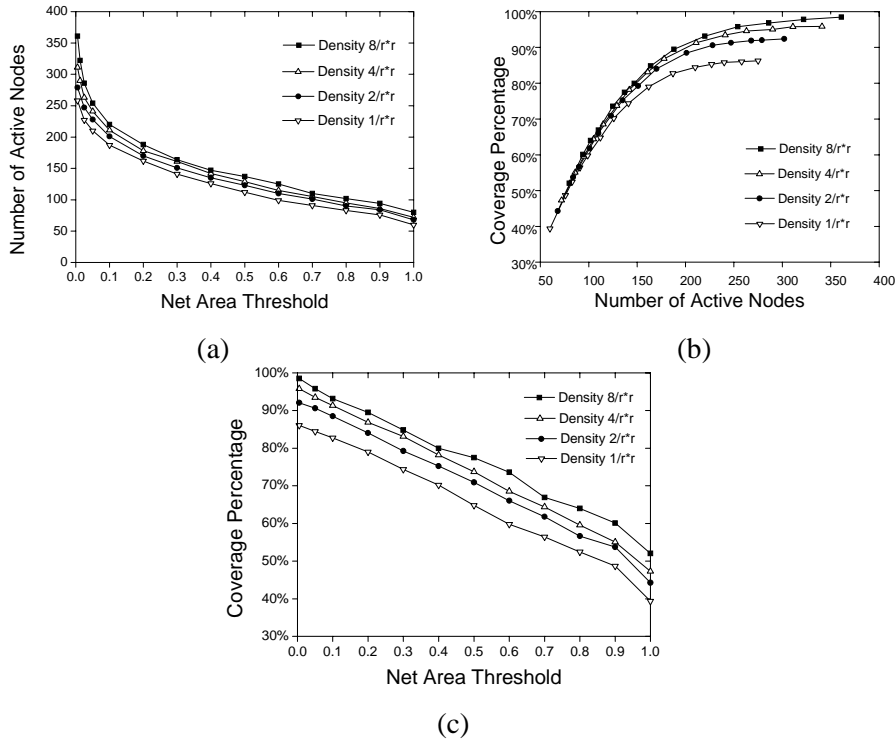


Fig.8. The coverage over node density using ACOS protocol

The energy dissipation is about in proportional to the number of active sensors. In Fig.8(a), we can see the number of active nodes ascends sharply as the net area threshold goes smaller. When $\varphi = 0$, all nodes are active, however, even if φ is a small non-zero number, the number of active nodes is much smaller than the total number of nodes. Fig.8(b) shows that the maximal coverage may be approached by much fewer active sensors than the total number. For example, when the node density is 8 per r-square, i.e. 3200 sensors in total, ACOS wakes up only 361 nodes but covers 98.5% of the whole region. Fig.8(c) illustrates that the coverage percentage is approximately linear to the net area threshold α . The coverage percentage can roughly be described as $(1-0.45\varphi) * maxC$, where $maxC$ is the maximum possible coverage of a given deployment.

5.3 Comparison

In this section, we implemented other three protocols RIS, PEAS and PECAS for comparison. In RIS, the time is divided into time slots of equal length T_{slot} at each sensor. Each T_{slot} is divided into two parts, the active period and the sleeping period. The duration of the active period is $p * T_{slot}$, where p depends on applications, and the sleeping period takes the rest part of a time slot. In PEAS, probing range R_p is given by the application depending on the degree of robustness it needs. In PEAS, a working node remains awake continuously until its physical failure or depletion of battery power. In PECAS, every sensor remains within the active mode for a duration indicated by parameter $Work_Time_Dur$ each time it wakes up.

5.3.1 Coverage

To compare all four protocols, ACOS, RIS, PEAS, and PECAS, we evaluate the coverage

over the roughly equal number of active sensors in the case of 800 sensors deployed. For each protocol, we take samples when the protocol runs stably after the initialization stage and before any sensor node depletes its energy.

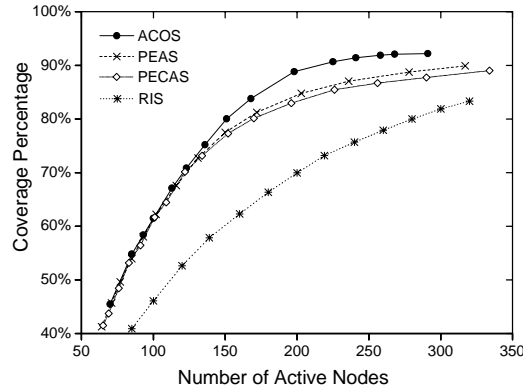


Fig.9. Coverage over the number of active sensors with 800 sensors deployed

As shown in Fig.9, for the same number of active nodes, ACOS can achieve more coverage than others. And as the number of active nodes increases, ACOS approximates the maximal coverage that can be achieved quite sooner than the other three protocols. RIS achieves much less coverage than other protocols, since each sensor independently follows its own sleep schedule and there is no collaboration between each other. PECAS achieves much close coverage to PEAS but a little less than it. This is caused by temporary sensing holes when a node has worked for a given time and went to sleep while other nodes cannot wake up right at the moment to fill the hole.

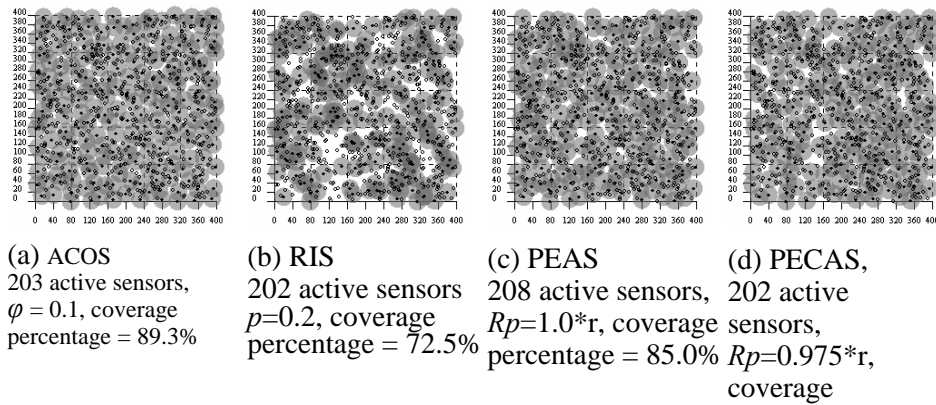


Fig.10. Spatial distribution of working sensors under different protocols

Fig.10 shows a typical scene of spatial distribution of working sensors under different protocols. The number of active nodes in each protocol is roughly 200. Fig.10(a) shows ACOS works quite well. There are no big holes and not much overlap. From the Fig.10(b) we can see that with the RIS protocol, there are many sensing holes and many sensors are densely clustered because of no collaboration. Fig.10(c) and Fig.10(d) show that PEAS and PECAS perform well but not as good as ACOS.

5.3.2 Network Lifetime

As analyzed in Section 5.2, when achieving equal coverage, RIS needs to keep much more nodes active compared to other protocols, so it consumes more energy than the others. PEAS keeps a little fewer nodes active compared to PECAS. But in PEAS, the working nodes will keep sensing continuously until its physical failure or depletion of power. The other nodes wake up and probe the environment, and then replace any failure nodes as needed. Thus, the density of functional sensors will drop down rapidly as time progresses. PECAS makes an enhancement to PEAS, a node changes to sleep mode after working for a given time period and the neighboring nodes wake up and turn to work. The coverage of the network with PECAS will drop more slowly than PEAS. Therefore, in the evaluation of coverage over time, we only compare our protocol with PECAS.

From Fig.9, we see that the performance of ACOS and PECAS are quite similar if the number of active sensors is small, such as less than 150 with 800 nodes deployed. The reason is that there are few redundant sensors when the number of active sensors is relatively small to the protected region. To measure the behaviors of each protocol exactly, we compare the protocols when more coverage is achieved. For example, the coverage percentage is more than 90% before any node dies. In this simulation, we set net Area Threshold $\varphi = 0.025$ for ACOS, and the probing range $R_p = 0.6 * r$ for PECAS. At the density 2 per r-square, i.e. 800 nodes deployed, for ACOS protocol, the average coverage percentage is 91.4%, and the average number of active nodes is 241 before any node dies. And for PECAS, the average coverage percentage is 91.2%, and the average number of active nodes is 405. Both the $Work_Time_Dur$ in PECAS and $T_{Wake_Duration}$ in ACOS are set to 2 hour. As mentioned in Section 5.1, we assume the battery of a sensor node can last for 100 hours before it dies.

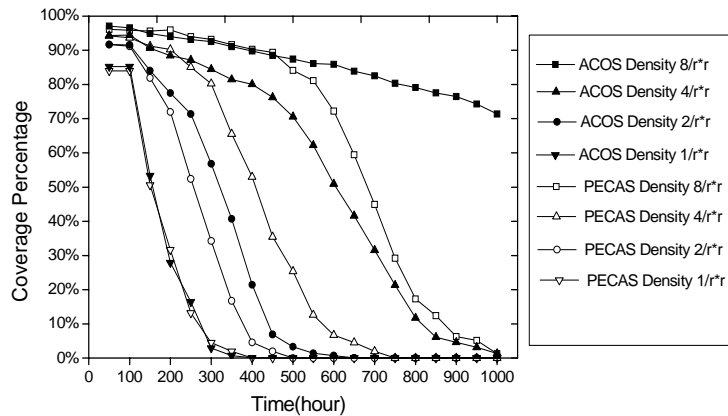


Fig.11. Coverage over time

From Fig.11, we see that over time with the same node density, the protected area is better covered with ACOS than with PECAS protocol in most time. For example, at density 4 per r-square, the coverage ratio with PECAS drops dramatically from the point of 300 hours, while the coverage ratio with ACOS still keeps in a relatively high level. Also, we can see that ACOS prolongs the network's lifetime. From the simulation results, we can see the two protocols perform much the same when the nodes are sparsely distributed, such as at the

density 1 per r-square. This is because there are few redundant sensors. As the density increases, ACOS performs better and better than PECAS. ACOS reduces redundancy than PECAS when they try to achieve nearly the same coverage. When the node density is 1 per r-square, the network lifetime of ACOS and that of PECAS is about the same. When the node density increases to 8 per r-square, the ratio of the network lifetime using ACOS (about 1700 hours) to the lifetime using PECAS (about 1000 hours) is approximately 1.7:1.

6. Conclusion

In this paper, we consider a fundamental problem of keeping sufficient coverage of the protected region while minimizing energy consumption and extending the lifetime of sensor networks. We have developed a sleeping protocol ACOS, which controls the mode of sensors to optimize the usage of energy as well as to maximize the coverage. We evaluate our protocol on a simulator and compare it with other sleeping protocols. The results demonstrate that our protocol has better coverage of the surveillance area while waking fewer sensors than other state-of-the-art sleeping protocols. The protocol extends the lifetime of sensor networks significantly and is excellent for highly dense sensor networks.

Acknowledgements

This research was supported partially by Natural Science Foundation of China grant #60442004.

References

- [1] G. Pottie and W. Kaiser. Wireless integrated network sensors, *Communications of the ACM*, 2000.
- [2] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson. Wireless sensor networks for habitat monitoring. In *First ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, 2002.
- [3] MICA2 Mote Datasheet.
http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/6020-0042-01_A_MICA_2.pdf.
- [4] Chao Gui and Prasant Mohapatra. Power Conservation and Quality of Surveillance in Target Tracking Sensor Networks. In *ACM MobiCom*, 2004.
- [5] Santosh Kumar, Ten H. Lai and József Balogh. On k-Coverage in a Mostly Sleeping Sensor Network. In *ACM MobiCom*, 2004.
- [6] F. Ye, G. Zhong, J. Cheng, S.W. Lu and L.X. Zhang. PEAS: a robust energy conserving protocol for long-lived sensor networks. In *the 10th IEEE International Conference on Network Protocols (ICNP)*, 2002.
- [7] Benyuan Liu and Don Towsley. A Study of the Coverage of Large-scale Sensor Networks. In *IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, 2004.
- [8] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. B.Srivastava. Coverage

- problems in wireless ad-hoc sensor networks. In *IEEE Infocom*, 2001.
- [9] S. Megerian, F. Koushanfar, G. Qu, and M. Potkonjak. Exposure in wireless sensor networks. In *ACM MobiCom*, 2001.
- [10] Santosh Kumar, Ten H. Lai and Anish Arora. Barrier Coverage With Wireless Sensors. In *MobiCom*, 2005
- [11] Y. Xu, J. Heidemann, and D. Estrin. Geography informed energy conservation for ad hoc routing. In *ACM MobiCom*, 2001.
- [12] B. Chen, K. Jamieson, and H. Balakrishnan. Span: An energy efficient coordination algorithm for topology maintenance in ad hoc wireless network. In *ACM MobiCom*, 2001.
- [13] A. Cerpa and D. Estrin. Ascent: Adaptive self-configuring sensor networks topologies. In *IEEE Infocom*, 2002.
- [14] S. Slijepcevic and M. Potkonjak. Power efficient organization of wireless sensor networks. In *IEEE Int'l Conf. on Communications (ICC)*, pages 472–476, 2001.
- [15] D. Tian and N. D. Georganas. A coverage-preserving node scheduling scheme for large wireless sensor networks. In *WSNA*, 2002.
- [16] Gang Lu, Narayanan Sadagopan, Bhaskar Krishnamachari, Ashish Goel. Delay Efficient Sleep Scheduling in Wireless Sensor Networks. In *Infocom*, 2005.
- [17] Mihaela Cardei, My T. Thai, Yingshu Li and Weili Wu. Energy-Efficient Target Coverage in Wireless Sensor Networks. In *Infocom*, 2005.
- [18] Hai Liu, Pengjun Wan, Chih-Wei Yi, Xiaohua Jia, Sam Makki and Pissinou Niki. Maximal Lifetime Scheduling in Sensor Surveillance Networks. In *Infocom*, 2005.
- [19] P. Bahl and V. N. Padmanabhan. RADAR: An in-building RF-based user location and tracking system. In *IEEE Infocom*, 2000.
- [20] Koen Langendoen, Niels Reijers. Distributed localization in wireless sensor networks: a quantitative comparison. *Computer Networks*, pages 499 - 518, 2003.
- [21] R. L. Moses, D. Krishnamurthy and R. M. Patterson. A self-localization method for wireless sensor networks. In *EURASIP J. Appl. Signal Process.*, pages 348-358, 2003.
- [22] Lingxuan Hu, David Evans. Localization for Mobile Sensor Networks. In *ACM MobiCom*, 2004.
- [23] Neal Patwari, Alfred O. Hero, III, Matt Perkins, Neiyer S. Correal and Robert J. O'Dea. Relative Location Estimation in Wireless Sensor Networks, *IEEE Trans. Signal Processing*, 2003.
- [24] C. Huang and Y. Tseng. The coverage problem in a wireless sensor network. In *WSNA*, 2003.