PEAS: A Robust Energy Conserving Protocol for Long-lived Sensor Networks

Fan Ye, Gary Zhong, Jesse Cheng*, Songwu Lu, Lixia Zhang
{yefan, gzhong, slu, lixia}@cs.ucla.edu
UCLA Computer Science Department, Los Angeles, CA 90095-1596

Abstract

In this paper we present PEAS, a robust energy-conserving protocol that can build long-lived, resilient sensor networks using a very large number of small sensors with short battery lifetime. PEAS extends the network lifetime by maintaining a necessary set of working nodes and turning off redundant ones. PEAS operations are based on individual node's observation of the local environment and do not require any node to maintain per neighbor node state. PEAS performance possesses a high degree of robustness in the presence of both node power depletions and unexpected failures. Our simulations and analysis show that PEAS can maintain an adequate working node density in the face of up to 38% node failures, and it can maintain roughly a constant overhead level under various deployment conditions ranging from sparse to very dense node deployment by using less than 1% of total energy consumption. As a result, PEAS can extend a sensor network's functioning time in linear proportion to the deployed sensor population.

Keywords: sensor networks, energy-conserving, robust network protocol

1 Introduction

Small, inexpensive sensors with constrained computing power, limited memory and short battery lifetime are coming into reality [7, 5]. When such nodes are deployed in an adverse environment that has high degrees of humidity, temperature, or even intentional destructions from malicious entities, in addition to node power depletion, unexpected node failures are likely to become norms rather than exceptions. Applications of sensor networks, on the other hand, desire a robust sensing system with extended life time. It is a great research challenge to build a resilient, long-lived sensor network with such small, fallible sensors.

This paper presents PEAS[1], a simple and distributed protocol that can build and maintain a resilient, long-lived sensor network out of large quantities of unreliable, short-lived sensor nodes. PEAS extends a network's functioning time by keeping only a necessary set of sensors in working mode and putting the rest into sleep mode. Sleeping nodes wake up once in a while to probe their neighborhood and replace any failed working nodes as needed. To be implementable on small sensors with stringent resource limitations, PEAS maintains a minimal amount of state at each node and involves very simple operations. Sensor nodes keep no per-neighbor node state, nor any information about the topology or lifetime estimation of their neighbors. When a node wakes up, it only needs to find out whether there exists any working neighbor within a local probing range to decide whether it should start working or go back to sleep. The wakeup frequency of sleeping nodes is self-adjusted to both maintain adequate working node density and minimize energy consumption. As shown by our analysis and simulation results, PEAS can extend a sensor network's functioning time in linear proportion to the number of deployed nodes, using less than 1% of the total energy consumption and withstanding up to 38% node failures.

Different from the protocols designed for ad-hoc networks which assume dynamic changes in connectivity but not frequent node failures, PEAS targets at a harsh working environment in which node failures may happen frequently. PEAS design also differs from existing energy-saving protocols, such as GAF[10], SPAN[4], ASCENT[3], and AFECA[9]. The above mentioned protocols are targeted for either ad-hoc networks or a relatively stable sensor network environment where nodes do not fail unexpectedly. Although they can all maintain a stable number of working nodes in the

*Jesse Cheng's email is jessec@ucla.edu

[1] PEAS stands for Probing Environment and Adaptive Sleeping.

presence of battery depletions, their operations either depend on the predictability of individual nodes' lifetime, or require each node maintain the state of all its neighbors. In contrast, PEAS assumes that the density of deployed nodes may be orders of magnitude higher than that of the working nodes, and that individual nodes may fail unexpectedly. These two assumptions make it infeasible to keep per neighbor node state or to reliably predict a node's lifetime.

The rest of the paper is organized as follows. We present the design of PEAS in Section 2 and analyze the conditions for asymptotic connectivity of PEAS in Section 3. We address several practical implementation issues in Section 4, and present the performance evaluation of PEAS in Section 5. Related work is discussed in Section 6, followed by the conclusion section. We would like to clarify that PEAS' role in a sensor network is to maintain a desired level of working sensor density to ensure both the sensing coverage and network connectivity. The actual sensing data delivery is carried out by a separate data forwarding protocol, such as those described in [11, 6].

2 PEAS Design

PEAS works with a sensor network consisting of a large number of inexpensive sensor nodes that can fail unpredictably. PEAS has two components: Probing Environment and Adaptive Sleeping. Probing Environment allows a newly wakeup node to probe its local neighborhood to discover whether a working node exists within a certain probing range. If no working node exists in that range, it starts working. Otherwise, it sleeps again. Adaptive Sleeping decides when a sleeping node should wake up again(or equivalently, the probing rate of each sleeping node). It ensures timely probing by sleeping nodes in a distributed manner. The goal is to make disruptions in sensing and communications (due to node failures) within what is tolerable by applications, while minimizing the probing overhead.

The designs of these two components are described in Sections 2.1 and 2.2, respectively. In the following, we assume that each sensor node may vary its transmission power and choose a power level to cover a circular area given a radius[2]. We discuss how PEAS works with fixed transmission power in Section 4.

[2]Some state-of-the-art hardware, e.g., MOTES, already allows variable transmission power [5].

2.1 Probing Environment

Each node in PEAS has three operation modes: *Sleeping, Probing* and *Working*. The state transition diagram among these three modes is shown in Figure 1.

Nodes are initially in the *Sleeping* mode. Each node sleeps for an exponentially distributed duration generated according to a probability density function (PDF) $f(t_s) = \lambda e^{-\lambda t_s}$, where λ is the *probing rate* of the node and t_s denotes the sleeping time duration.

After a node wakes up, it enters the *Probing* mode. A probing node seeks to detect whether any working node is present within a certain probing range R_p. The probing node uses an appropriate transmission power to broadcast a PROBE message within its local probing range R_p. Any working node(s) within that range should respond with a REPLY message, also sent within the range of R_p. It is possible that multiple working nodes exist within R_p when a node probes. To reduce collisions, each working node waits for a small random period before it sends back the REPLY.

If the probing node hears a REPLY, it goes back to the *Sleeping* mode for another random period of time t_s, generated according to the same PDF. But λ is adjusted according to the Adaptive Sleeping algorithm in Section 2.2 based on the feedback information carried in the REPLY. If the probing node does not hear any REPLY, it enters the *Working* mode and starts functioning until it fails or consumes all its energy.

Figure 2 gives a simple example for illustration. At time t_1, nodes 2 and 3 are in the working mode. Node 1 wakes up and broadcasts a PROBE message within a probing range R_p. Because no working nodes exist within R_p, node 1 starts working. At time t_2, sleeping node 4 wakes up and probes. Because node 2 is within node 4's probing range, it responds with a REPLY message. Upon hearing the REPLY, node 4 sleeps again. Then node 2 dies at time t_3, and node 6 wakes up at time t_4. After probing, node 6 starts working and replaces node 2.

The initial value of λ decides how quickly the network acquires enough number of working nodes during the boot-up phase. For instance, 50% of the deployed nodes are required for the network to function and the application requires the network start functioning 1-minute after deployment. Based on the PDF, we can calculate that an initial λ of 0.012 ensures that 50% of the nodes wake up at least once within the first minute after deployment. Since PEAS adjusts the probing rates, we may choose a higher λ to ensure a fast-functioning network.

The probing range R_p determines the redundancy

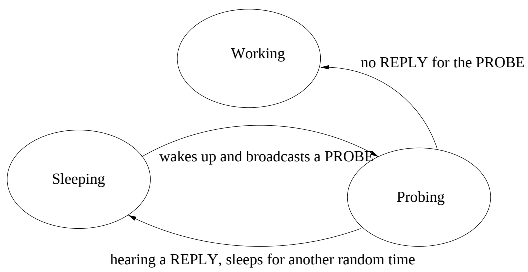

Figure 1. State Transition Diagram of Operations at Each Node

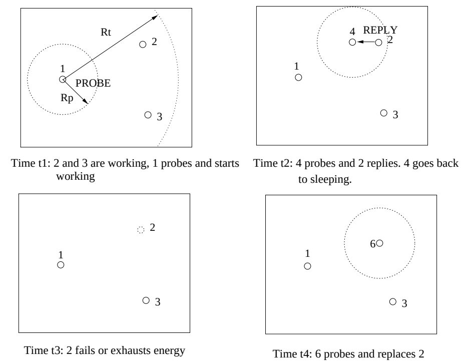

Figure 2. An Example of Probing Environment

of working nodes. It is specified by the application based on its requirements for both robust sensing and robust communicating. These two functions may require different densities of working nodes. For example, a type of sensors can detect animals within 10 meters and transmit up to 20 meters. Suppose an application decides that working nodes should be spaced at most 3 meters for robust sensing, but 6 meters are enough for robust communication. The application may simply choose the probing range R_p as the smaller value of 3 meters[3]. The choice of R_p also affects network connectivity; this is to be analyzed in Section 3.

2.1.1 Design rationale

We make two important decisions in the design of Probing Environment: (1) a node's location decides whether it should be turned on or not, and (2) the sleeping time of a node is randomized. We now explain the rationale.

Location-dependent working nodes Unlike other related schemes that choose to turn on nodes with more energy or more neighbors [10, 4], PEAS does not favor such nodes and treats all of them equally. This is motivated by the sensor network characteristics. In a sensor network built by unreliable, densely distributed nodes, it is the number, not the capability of each individual node that really matters. The system relies on the *collective* behavior of nodes to function reliably. As long as PEAS maintains enough working nodes, they can perform required sensing and communicating tasks.

Location-based probing rule also ensures that adjacent working nodes be placed at an appropriate distance, which allows for desired redundancy to guarantee resilient sensing and communicating functions. This feature is important because overly dense working nodes not only increase collisions, but also unnecessarily waste precious energy resources. Whereas too sparse working nodes may not satisfy the required degree of redundancy, e.g., certain areas may be left unmonitored.

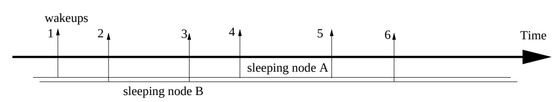

Figure 3. In PEAS, sleeping nodes A and B have randomized sleeping times

Randomized sleeping time A node in PEAS sleeps for a randomized period of time. The wakeups of nodes are spread over time (shown in Figure 3). This is different from the related schemes [10, 4] which typically take the deterministic approach of synchronized sleeping and waking-up: All sleeping nodes (in a local neighborhood) doze for the same predicted period of time, which is normally their working neighbors' active time. Then they all wake up almost simultaneously to re-elect new working nodes.

Such a deterministic approach is feasible only if its intended environment is predictable (i.e., the lifespan for a working node can be reliably estimated beforehand), which again depends on the assumption of reliable nodes. In a harsh environment with unreliable sensors, the predictability of a node's lifespan no longer holds. When a working node fails unexpectedly before its expected lifespan, there come large "gaps" in the system during which no working node is available (illustrated in Figure 4).

Therefore, PEAS chooses to distribute node wakeups over time, rather than to cluster them at a few time instants. Shown in Figure 5, node wakeups come at

[3]Designing sensor hardware that balances these two functions is not the topic of this paper. We expect hardware developers to address this issue.

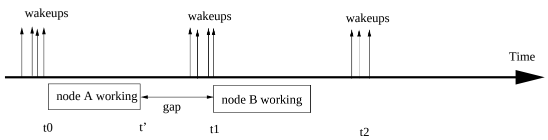

Figure 4. Synchronized operation has big gaps when nodes fail

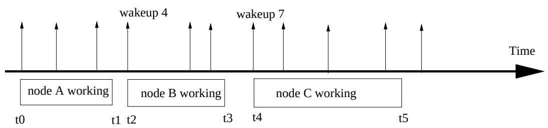

Figure 5. Distributing wakeups over time shortens gaps

much shorter time intervals. Thus the average gap between two successive working nodes in any local neighborhood can be greatly shortened. The spreading also reduces collisions incurred by synchronized wakeups.

A remaining question about Probing Environment is why it uses exponential distribution for the random sleeping time. We will show in Section 2.2 that exponential distribution leads to a Poisson process of probing events; this exhibits nice properties that the Adaptive Sleeping is built upon.

2.2 Adaptive Sleeping

Adaptive Sleeping adjusts the probing rate λ of each sleeping node. The goal is to keep the aggregate probing rate $\bar{\lambda}$ of all the sleeping neighbors of each working node at about a desired rate λ_d, which is specified by the application. This way, the transient interruptions in sensing and communication are acceptable to the application, while keeping the probing frequency in check.

The design issue is that, the number of sleeping neighbors of a working node changes over time, varies in different locations, and λ has to be adjusted dynamically to adapt to such varying conditions. The basic idea is to let each working node measure the aggregate probing rate $\bar{\lambda}$ it perceives from all its sleeping neighbors. The working node then includes the measured rate $\bar{\lambda}$ when sending a REPLY message to a probing neighbor. Each probing node then adjusts its probing rate λ accordingly to generate a new sleeping time period. The details are as follows.

Measuring aggregate $\bar{\lambda}$ at a working node Each working node maintains two states:

- N: a counter that records how many PROBEs have been received.
- t_0: the most recent time when N is set to 0.

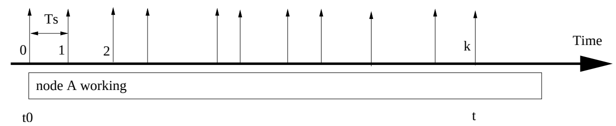

Figure 6. Measure $\bar{\lambda}$

When the working node hears the first PROBE message, it sets the counter to 0, and t_0 to the current time t. After that, each time a new PROBE is received, the counter increments by one. Eventually when the counter reaches a threshold value k (k is set to 32 as we explained in Section 2.2.1), a measurement $\hat{\lambda}$ of the actual probing rate $\bar{\lambda}$ is calculated as follows[4]:

$$\hat{\lambda} = \frac{k}{t - t_0}, \tag{1}$$

where t is the current time. The node then sets t_0 to t, resets the counter to 0, and repeats the above process (see Figure 6 for an illustration). Whenever a working node receives a PROBE message, it includes its current probing rate measurement $\hat{\lambda}$ and the desired probing rate λ_d in its REPLY message.

Adjusting per-node probing rate λ at each probing node Upon receiving a REPLY message from the working node, the probing node updates its current probing rate λ based on the received $\hat{\lambda}$:

$$\lambda^{new} = \lambda \frac{\lambda_d}{\hat{\lambda}}. \tag{2}$$

Then the probing rate will use λ^{new} to generate a new sleeping period t_s according to the probability density function $f(t_s) = \lambda^{new} e^{-\lambda^{new} t_s}$.

2.2.1 Explanation

We now explain why the above algorithm keeps the aggregate $\bar{\lambda}$ around the desired λ_d. From the probability theory [8], the exponentially distributed intervals between successive wakeups observe a Poisson process of wakeup events. Probings from different sleeping neighbors still construct a Poisson process, but with a parameter $\bar{\lambda}$, the sum of all sleeping nodes' rates λ_i:

$$\bar{\lambda} = \sum_{i=1}^{n} \lambda_i, \tag{3}$$

[4] We also tried a moving average measurement, but the choice we present here turned out to work better.

where n is the number of sleeping neighbors and λ_i is the probing rate of the ith neighbor.

We utilize the property of Poisson processes to measure $\bar{\lambda}$. It is known that the average interval $\bar{T_s}$ of the aggregate Poisson process is given as $\bar{T_s} = \frac{1}{\bar{\lambda}}$. By measuring the average interval $\bar{T_s}$, we can derive the aggregate rate $\bar{\lambda}$. This is exactly what (1) does.

To obtain an accurate estimate $\hat{\lambda}$ that is close to the actual $\bar{\lambda}$, the constant k in (1) has to be large enough. Because the intervals are i.i.d. random variables, we apply the central limit theorem [8] to estimate how large k should be for a reasonably good measurement. It turns out that when $k > 16$, with over 99% confidence the measured average has only 1% error compared with the real value. We select $k = 32$ based on experimental studies. This also accounts for the short random time each working node waits before sending its REPLY and the latency in communication and message processing.

Assume that the measured rate $\hat{\lambda}$ is accurate, i.e., $\hat{\lambda} \approx \bar{\lambda}$. After each sleeping neighbor adjusts its probing rate according to (2), the new aggregate probing rate becomes

$$\bar{\lambda}_{new} = \sum_{i=1}^{n} \lambda_i^{new} = \sum_{i=1}^{n} \lambda_i \frac{\lambda_d}{\hat{\lambda}} \approx \frac{\lambda_d}{\bar{\lambda}} \bar{\lambda} = \lambda_d.$$

Thus the aggregate probing rate reaches the desired rate λ_d.

The above derivation is idealistic since it assumes that all sleeping nodes hear the measurement and adjust their rates on time. In practice, if some nodes sleep for longer periods and miss the current measurement, they may receive a different measurement. Thus $\bar{\lambda}$ may not be the same as λ_d. But as long as the working node keeps measuring and feeding-back this information, $\bar{\lambda}$ should be fluctuating around λ_d. We further evaluate the effectiveness of Adaptive Sleeping in Section 5.

It is possible to calculate $\bar{\lambda}$ directly by using (3) (a working node sums up all λ_i directly). However, this poses the difficulty of keeping per-neighbor state λ_i. Due to unexpected failures and potentially dense deployment, a working node may not know precisely how many sleeping neighbors it has. Thus it does not know when it has collected *all* λ_is for its sleeping neighbors. In addition, if some neighbor fails during sleeping, the working one does not know whether it is because the node has failed, or because it has a very long sleeping period. Hence, it cannot decide whether the corresponding λ_i should be kept or removed.

In the design, we intentionally make design choices that trade-off optimality for simplicity and better robustness. This is why in Adaptive Sleeping no node keeps per-neighbor state. A more complex design may include per neighbor states to optimize the energy consumption. However, as long as the required connectivity and coverage are satisfied, we opt for the simplest design to make PEAS robust and implementable on very small nodes.

A final comment is that the desired probing rate λ_d should be given by the application and depends on the application's tolerance of interruptions in sensing and/or communication. For example, if an animal-tracking sensor network allows for monitoring interruptions up to 5 minutes, λ_d can be set at 1 per 300 seconds to ensure that the lengths of "gaps" in sensing are acceptable to the application.

3 Asymptotic Connectivity of PEAS

A PEAS model We present the following PEAS model to aid the analysis. Consider a two-dimensional network field[5]. We imagine each working node as a *round* pea that occupies a circular area of radius $R_p/2$. Sleeping or probing nodes do not occupy any area. The distance between the centers of any two peas is at least $R_p/2 + R_p/2 = R_p$, which holds true when two peas are tangent to each other.This is exactly what the probing rule produces. On the other hand, any two working nodes are separated by a distance of at least R_p. Therefore, positioning of working nodes is *equivalent* to the placement of peas on the plane.

To find out the conditions under which PEAS ensures a.a.s. connectivity, let us consider a sufficiently large network $R = [0, l]^2$ that is divided into square cells, each of which is of size $c \times c$ and $c = R_p$. We first derive the conditions under which each cell has at least one node a.a.s. based on Blough's Theorem 2 in [2]. We then show that, if each cell has at least one node, PEAS ensures connectivity a.a.s. when the maximum transmitting range $R_t \geq (1 + \sqrt{5})R_p$.

The following Lemma 3.1 specifies the condition under which each cell has at least one node a.a.s. The complete proof is similar to that of Theorem 2 in [2]. Due to space limit we put it in a technical report [12].

Lemma 3.1 *Consider the case when n nodes are uniformly distributed in $R = [0, l]^d$ for $d = 2$, and assume that $c^d n = k l^d \ln l$ for some constant $k > 0$. Let $\mu_0(n)$ be the random variable denoting the number of empty cells. If $k > d$, then $\lim_{l \to \infty} E[\mu_0(n)] = 0$, where $E[\mu_0(n)]$ is the expected number of empty cells.*

Given the above condition, we have

Lemma 3.2 *When Lemma 3.1 holds, i.e., each cell has at least one node a.a.s., for any*

[5] The model applies to three-dimensional as well.

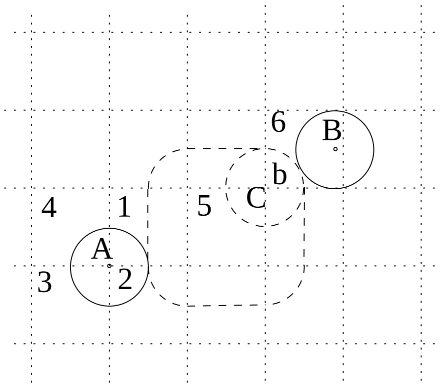

Figure 7. The minimum distance between two adjacent working nodes

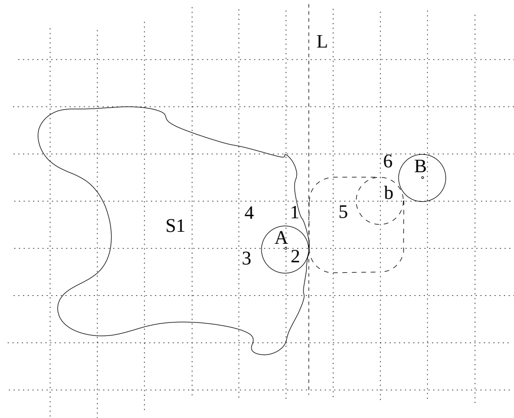

Figure 8. A "disconnected" component always has another working node connected

working node A and its working neighbor B, $\lim_{l \to \infty} P(min(Dist(A,B)) < (1+\sqrt{5})c) = 1$, where $Dist(A,B)$ denotes the distance between A and B, and $min(Dist(A,B))$ is the distance between A and the closest working neighbor.

Proof Because Lemma 3.1 holds a.a.s. no matter how the grid is oriented or where the grid is positioned, without any loss of generality, we let node A be at the center of cells 1, 2, 3 and 4 (Figure 7). According to the peas model, each working node is a round pea of radius $c/2$ and peas do not overlap with each other. To avoid obscuring the main idea, we will consider boundary cases later.

Consider the worst-case scenario in which all other working nodes are as far away from node A as possible. In such cases, other nodes in cells 2, 3 and 4 are all within the probing range of A, and they are all sleeping. Consider node C in cell 5, which is to the right of cell 1. Give that node C is uniformly distributed and can be anywhere in cell 5, the farthest it can be from A is at the upper right corner b of cell 5.

In order to put node C (centered at corner b) into the sleeping mode, node B must be working within the probing range of C. Based on geometry calculations, the farthest position where B can be from node A, is in cell 6 and with a distance of $(1+\sqrt{5})c$. This is the minimum distance within which there must exist at least another working node. Otherwise, if all working neighbors are farther than this minimum distance away, node C will always be working, no matter where it is located within cell 5.

We next consider the boundary case. The number of nodes in boundary cells is $O(l)$, which is an order of magnitude lower than the total number $O(l^2)$. Therefore, it follows that $P(min(Dist(A,B)) < (1+\sqrt{5})c)$ still approaches 1 as $l \to \infty$. ∎

Now we derive the condition for asymptotic connectivity in PEAS.

Theorem 3.1 *If the transmitting range $R_t \geq (1+\sqrt{5})R_p$, and the conditions in Lemma 3.1 are satisfied, then $\lim_{l \to \infty} P_{conn}(PEAS) = 1$, where $P_{conn}(PEAS)$ denotes the probability that working nodes in PEAS are connected.*

Proof We prove the theorem by contradiction. Suppose the working nodes are not connected. Let us consider a connected component S_1 formed by a subset of working nodes. Any working node in S_1 has working neighbors only in S_1. Without any loss of generality, we consider the "rightmost" node A in S_1, and draw a grid that is centered with A at a crossing point. A vertical line L is tangent to A (Figure 8). Any pea (working node) in S_1 is to the left of L. Using arguments similar to Lemma 3.2, there must be a working node B which is to the right of L and has at most a distance of $(1+\sqrt{5})c$ to A. Since the transmitting range $R_t \geq (1+\sqrt{5})R_p$ and $R_p = c$, nodes A and B are connected. This contradicts the assumption that any working node in S_1 is only connected to other nodes in S_1. Therefore, there is no such a disconnected component. Note that based on similar reasoning, the boundary case does not affect the asymptotic connectivity as $l \to \infty$. This completes the proof. ∎

4 Discussions

Compensate packet losses Due to collisions, PROBE and REPLY messages may get lost and cause probing nodes to work unnecessarily. To reduce such errors, we let a probing node transmit multiple PROBEs thus each working node reply multiple times. These multiple messages are randomly spread over a small time interval to reduce collisions. In experiments we found that three PROBEs work well against loss rates of up to 10%. These multiple messages will increase energy but our evaluation in Section 5 shows that the energy overhead is still smaller than 1%.

To further correct such errors once they happen, we can make unnecessary working nodes go back to sleep: Each working node reacts to REPLYs sent by its working neighbors, which respond to probing nodes. Because REPLYs are also sent within a distance of R_p, two working nodes are less than R_p away if they can hear the REPLYs from each other. We can let one of them go back to sleep. In practice, if either of the two working neighbors can turn off the other, they may take turns to work and cause an unstable working node topology. Since many routing protocols have to rebuild routing states in new working nodes and suffer from unstable topologies, we favor the one that has been working for a longer time to stablize the topology: Each working node records the time when it starts working. It calculates and includes the time T_w – how long it has been working – in its REPLYs. When a working node hears a REPLY, it goes to sleep only if its T_w is less than that of the sender's. So nodes that have been working for longer times can turn off new working ones, but not vice versa.

Probing nodes with more than one working neighbors When a probing node has more than one working neighbors within its probing range, it may hear several REPLY messages, each of which contains a different $\bar{\lambda}$. Such a node cannot replace any of the working neighbors alone because it can work only when all such working neighbors die. The probing from this node is not critical to the replacement of its working neighbors. We simply let such a probing node adjust its λ according to the largest measurement value, resulting in the lowest probing rate.

Nodes with fixed transmission power In Section 2 we assume that each node can choose a transmitting power to reach a desired probing range R_p. For sensors with fixed transmission power, they can use a threshold filtering rule regarding the received signal strength. A working node reacts only to PROBE messages with signal strengths greater than a threshold S_{th}. Similarly, a probing node goes to sleep again only if the REPLY has a signal strength greater than S_{th}. In a harsh environment, irregularities in signal attenuation may generate different signal strengths in different areas, thus working nodes in areas with poorer signal reception can be denser than those in other areas. We believe that this is desirable because it is only with more working nodes in such areas that the same level of robustness is maintained.

Distribution of deployed nodes As long as the deployed nodes are dense enough everywhere, PEAS can keep enough working nodes in each region, independent of the particular distribution of deployed nodes. But the deployment distribution of sensor nodes does affect the performance of the network. An uneven distribution may cause the system to function for less time because regions with fewer nodes will die out much earlier. This argues for evenly distributed sensor deployment. Although a complete study of this issue is out of the scope of this paper, we believe evenly deployed nodes will work longer than those deployed irregularly.

5 Performance Evaluation

5.1 Methodology and Metrics

We implement PEAS in PARSEC [1] and select sensor hardware parameters similar to Berkeley Motes [5]. The node power consumptions in transmission, reception, idle and sleep modes are 60mW, 12mW, 12mW and 0.03mW, respectively. The initial energy of a node is randomly chosen from the range of $54 \sim 60$ Jules to simulate the variance of battery lifetime, allowing the node to operate about $4500 \sim 5000$ seconds in reception/idle modes. The sensing and maximum transmitting ranges are both 10 meters. Each node has a raw wireless communication capacity of 20Kbps. The packet size of PROBE and REPLY messages is 25 bytes, which is enough to hold the information they need to carry.

5.2 Prolong system functioning time

We use a $50 \times 50m^2$ network field, and nodes are uniformly distributed in the field initially and remain stationary once deployed. A source and a sink are placed in opposite corners of the field. The source generates a data report every 10 seconds and the data report is delivered to the sink using the GRAB forwarding protocol [11]. The initial per-node probing rate λ is chosen as 0.1 wakeup/sec so that the number of working nodes quickly stabilizes. The probing range is set to 3 meters. The desired aggregate probing rate λ_d is chosen as 0.02 wakeup/sec, which is equivalent to a wakeup every 50 seconds perceived by a working node.

To evaluate the robustness of PEAS protocol, we artificially inject node failures which are randomly distributed over time in the simulation. The *failure rate* denotes the average number of failures per unit time.

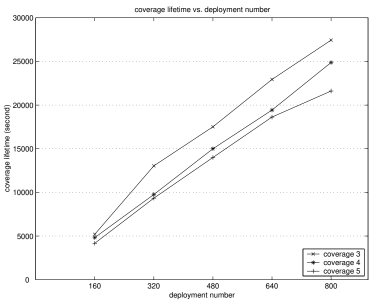

Figure 9. Extension of Coverage Lifetime

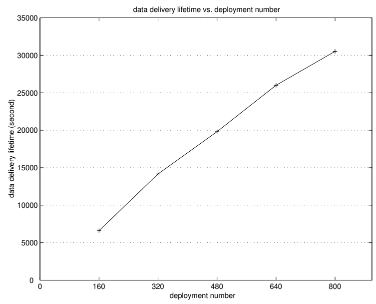

Figure 10. Extension of Data Delivery Lifetime

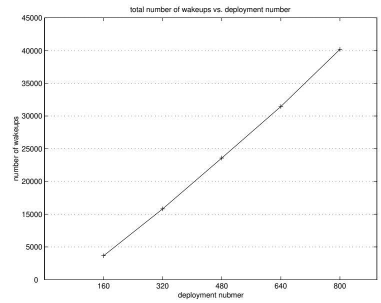

Figure 11. Average Total Wakeup Count for Deployment Numbers

The *failure percentage* is the percentage of failed nodes. Note that failures are deaths not incurred by energy depletions.

The main metrics used are *sensing coverage lifetime* and *data delivery lifetime*. The *sensing coverage* is defined as the percentage of the field monitored by working nodes. An application may require that each point in the field be monitored by at least K working nodes for robustness. We define *K-coverage* (or coverage K) as the percentage of the field size monitored by at least K working nodes. The lifetime of K-coverage is the time duration from the beginning until K-coverage drops below a threshold value. It characterizes how long the system ensures that interested events are monitored and reported properly.

The *data success ratio* at any time is the ratio of the number of reports successfully received at the sink to the total number of reports generated by the source up to that time. *Data delivery lifetime* is defined as the time when the data success ratio drops below a threshold. It specifies how long the network can deliver reports to users. Both threshold values are chosen as 90%.

We measure the overhead of PEAS by the number of wakeups and calculate the energy consumed by its operations.

To see how PEAS adapts to varying node population, we set the node number as 160, 320, 480, 640 and 800 and simulate for a sufficiently long period of time until all nodes die. We assume the application requires that each point be monitored by at least 4 working nodes. Given a probing range of 3 meters, 160 nodes result in a close to 100% 4-coverage ratio but less than 95% 5-coverage, so we choose 160 as the "base" number. Given each node population, the results are averaged over 5 simulation runs, each of which uses a failure rate of 10.66 failures/5000 seconds.

We now present how PEAS extends the coverage and data delivery lifetimes with more deployed nodes.

Node Number	Energy Overhead	Overhead Ratio
160	11.58J	0.143%
320	34.18J	0.207%
480	58.68J	0.236%
640	83.53J	0.25%
800	111.11J	0.267%

Table 1. Energy Overhead for Deployment Numbers

Figure 9 shows the lifetimes of 3, 4 and 5-coverage. As the sensor population increases, each lifetime increases almost linearly. This is because PEAS keeps only a necessary number of nodes working, while turning off others. The more deployed nodes, the more in the sleeping mode, and the longer they can keep the sensing coverage. We also observe that the lifetimes of 3-coverage are longer than those of 4-coverage, because less working nodes are required to cover each area by at least 3 nodes. Similar is true for the cases of 4- and 5-coverage.

The data delivery lifetime is shown in Figure 10. Given 160 nodes, the data delivery lifetime is about 6600 seconds, longer than the maximum idling lifetime of a node. This is because the working nodes that replace the initial set of working ones still deliver some reports. So it takes some time after 5000 seconds for the total success ratio to drop below the 90% threshold value.

As the deployment number increases, the average data delivery lifetime increases linearly. Each additional increase in node number prolongs the delivery lifetime for about another 6000 seconds. The above results demonstrate that PEAS is able to increase the network functioning lifetime (sensing and communicating) in proportional to node population.

We then look at the overhead incurred for PEAS' operations. Figure 11 shows the average number of wakeups for each deployment number. This number also grows linearly as the node population increases.

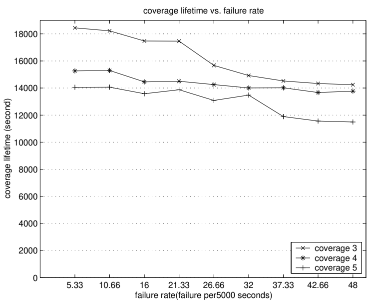

Figure 12. Coverage Lifetime with Failures

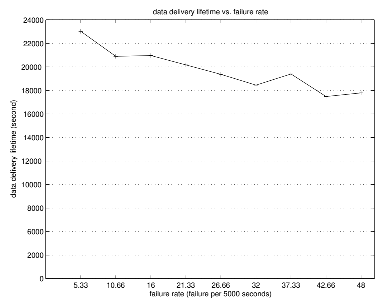

Figure 13. Data Delivery Lifetime with Failures

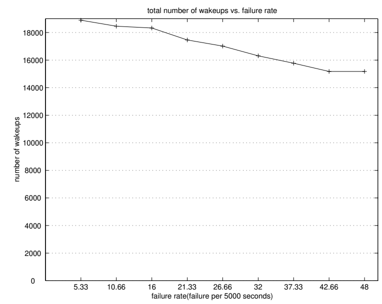

Figure 14. Average Total Wakeup Count for Failure Rates

This is because Adaptive Sleeping adjusts the wakeup frequency to the desired level. When the network functions longer, more wakeups happen.

To measure the energy overhead, we first calculate the energy used in each wakeup. The energy used in a wakeup consists of the amount in transmitting and receiving PROBE and REPLY messages, and the amount a probing node waits in idling to receive REPLYs. Based on the current implementation in which a probing node transmits three PROBEs and waits for 100ms during which working nodes randomly back off to send REPLYs, the amount is 0.00316 Joule per wakeup. Using this estimation, we plot the amount of energy overhead and its ratio compared with the total energy consumption in Table 1. The table shows that the energy overhead is less than 0.3% of the total energy consumption. This small energy overhead demonstrates the efficiency achieved by the simplicity and adaptivity of PEAS.

5.3 Robustness against node failures

We now evaluate the robustness of PEAS against node failures. The initial node population is set to 480 and we increase the failure rate from 5.33 to 48 failures per 5000 seconds at incremental steps of 5.33. We calculate the average failure percentage — the ratio of failed nodes to the total deployed nodes and find that there are about 38% nodes that fail in the maximum failure rate case.

Similarly, we use the coverage and data delivery lifetimes to evaluate the robustness of PEAS. If PEAS is not robust enough to maintain sufficient working nodes in the presence of severe node failures, the system would work for disproportionately less time or might not function at all.

Figure 12 plots the coverage lifetimes under the failure rates from 5.33 to 48. As the failure rate increases, system lifetime tends to decrease. However, as long as there are enough sleeping nodes to overcome node failures, PEAS maintains a high coverage above the threshold. Even with the most severe failures (with 38% node failure), the coverage lifetime drops only between 12% to 20%. This shows that not only failed nodes are replaced, but also the replacements happen quickly enough to minimize interruptions (The few abnormal points were caused by random factors).

The average data delivery lifetime for each failure rate is shown in Figure 13. The drop is about 20%, similar to that of coverage lifetime. This shows that PEAS maintains enough working nodes to provide high quality communication connectivity in the presence of severe node failures.

Finally we present the wakeup and energy overhead. The robustness of a protocol should not come at the cost of excessive overhead to combat failures. For PEAS, the number of wakeups decreases as the failure rate increases (Figure 14). This is because there are less sleeping nodes for higher failure rates. We also measure the energy overhead for all failure rates, and it is constantly less than 0.25% of the total energy consumption. The same level of small overhead for varying failure rates demonstrates that PEAS achieves robustness at roughly constant overhead and does not consume excessive energy to overcome failures.

6 Related Work

To preserve the limited battery power, various approaches have been explored to put unnecessary nodes into sleeping mode for both wireless ad hoc networks (e.g. GAF[10], SPAN[4], AFECA[9]) and sensor networks (e.g. ASCENT[3]). SPAN lets each node keep a list of all its working neighbors and exchange this list with its neighbor nodes. As a result, all nodes learn the connectivity within their 2-hop neighborhood to decide which nodes to turn off. The sleeping nodes wake up at a scheduled time interval to re-elect working ones. GAF divides the network into grid cells. Assuming each node knows its location through GPS or other location service, each node knows which grid it is in. Within each grid only one node stays up and the rest goes to sleep, with the sleep time set as a function of the remaining energy of the working node. As a result, the number of wakeups in GAF is proportional to the number of deployed nodes. In AFECA, each node maintains a list of neighbor identifiers in order to keep track of the number of neighbors, based on which it decides the sleeping period. In ASCENT, each node measures the number of active neighbors and per-link data loss rates through data traffic. The node decides whether it should work or go to sleep based on a function of the above factors. In summary, the above mentioned solutions either maintain some per-neighbor node state at each node, or operate based on the predicted lifetime of the working nodes (or do both).

To prolong the system lifetime PEAS uses the same basic approach of turning off unused nodes to preserve energy. However to be both resilient against unpredictable node failures in a harsh or even hostile environment and versatile under various degrees of deployment density, in PEAS nodes do *not* keep any per-neighbor information. PEAS utilizes a random wakeup algorithm that adapts to observed node failure rate. Instead of counting the number of neighbor nodes, a wakeup node in PEAS probes the *space* surrounding itself to decide whether to go back to sleep. Instead of relying on multiple node coordination such as an election to decide who should be the working node, PEAS design exploits randomization and adaptivity to achieve simplicity and scalability.

7 Conclusion

Although it is economically feasible to build large-scale sensor networks using large quantities of inexpensive sensors, building a resilient, long-lived network with unreliable, short-lived sensors remains a research challenge. Since a real operational environment may be harsh or hostile, nodes may fail unexpectedly before their power depletion. Existing energy-saving protocols have mostly focused on maintaining a stable set of working nodes in the presence of node power depletion, without paying attention to *unexpected* node failures.

We have developed PEAS, a distributed and randomized energy-saving protocol for sensor networks. PEAS lets each node probe its local operating *space* to maintain a desired working node density while avoiding the overhead of keeping per neighbor state. To handle unpredictable node failures, PEAS uses a randomized wakeup algorithm that can self-adapt to node failures. PEAS keeps the working node density approximately constant independent of the node deployment density with roughly the same overhead. As a result, PEAS can prolong the overall system lifetime proportionally to the total number of deployed nodes. Our analysis and simulation results confirmed the effectiveness of the design.

References

[1] *Parallel Computing Laboratory, Computer Science Department, UCLA*, http://pcl.cs.ucla.edu/projects/parsec/.

[2] D. M. Blough and P. Santi. Investigating Upper Bounds on Network Lifetime Extension for Cell-Based Energy Conservation Techniques in Stationary Ad Hoc Networks. *MOBICOM 2002*.

[3] A. Cerpa and D. Estrin. Ascent: Adaptive self-configuring sensor networks topologies. *In Proceedings of the Twenty First International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2002), New York, NY, USA*, June 23-27 2002.

[4] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris. SPAN: An Energy Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks. *MOBICOM 2001*.

[5] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System Architecture Directions for Networked Sensors. *International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-IX)*, 2000.

[6] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks. *ACM International Conference on Mobile Computing and Networking (MOBICOM'00)*, 2000.

[7] J. Kahn, R. Katz, and K. Pister. Next Century Challenges: Mobile Networking for "Smart Dust". *ACM International Conference on Mobile Computing and Networking (MOBICOM'99)*, 1999.

[8] S. Ross. Introduction to Probability Models, 6th Edition. *Acedemic Press*, 1997.

[9] Y. Xu, J. Heidemann, and D. Destrin. Adaptive Energy-Conserving Routing for Multihop Ad Hoc Networks. *USC/ISI Research Report 527*, October, 2000.

[10] Y. Xu, J. Heidemann, and D. Estrin. Geography Informed Energy Conservation for Ad Hoc Routing. *ACM International Conference on Mobile Computing and Networking (MOBICOM'01)*, 2001.

[11] F. Ye, G. Zhong, S. Lu, and L. Zhang. A Robust Data Delivery Protocol for Large Scale Sensor Networks. *The 2nd International Workshop on Information Processing in Sensor Networks (IPSN '03)*, 2003.

[12] F. Ye, G. Zhong, S. Lu, and L. Zhang. A Robust Energy Conserving Protocol for Long-lived Sensor Networks. *UCLA CSD Technical Report*, 2003.