Wireless Netw
DOI 10.1007/s11276-006-6058-8

A randomized energy-conservation protocol for resilient sensor networks*

Fan Ye · Honghai Zhang · Songwu Lu · Lixia Zhang · Jennifer Hou

Published online: 27 April 2006

Abstract In this paper we present PEAS, a randomized energy-conservation protocol that seeks to build resilient sensor networks in the presence of frequent, unexpected node failures. PEAS extends the network lifetime by maintaining a necessary set of working nodes and turning off redundant ones, which wake up after randomized sleeping times and replace failed ones when needed. The fully localized operations of PEAS are based on each individual node's observation of its local environment but do not require per neighbor state at any node; this allows PEAS to scale to very dense node deployment. PEAS is highly robust against node failures due to its simple operations and randomized design; it also ensures asymptotic connectivity. Our simulations and analysis show that PEAS can maintain an adequate working node density in presence of as high as 38% node failures, and a roughly constant overhead of less than 1% of the total energy consumption under various deployment densities. It extends a sensor network's functioning time in linear proportional to the deployed sensor population.

Keywords Sensor networks · Energy-conserving · Robust network protocol

1. Introduction

Advances in communication, computation and sensing have brought into reality small, inexpensive sensors such as Berkeley motes [1] and single-chip specs [2]. These sensors typically have constrained computing power, small memory space, and limited battery energy. When such nodes are deployed in an adverse environment with high degrees of humidity, temperature, or even intentional destructions from malicious adversaries, unexpected node failures are likely to happen in addition to the more predictable failures due to energy depletion; the occurrence of node failures may become norms rather than exceptions of the network. Sensor network applications, on the other hand, desire a robust sensing system with extended life time. Building a resilient, long-lived sensor network with such small, fallible sensors poses a great research challenge.

This paper presents PEAS (Probing Environment and Adaptive Sleeping), a simple, 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 the working mode and putting the rest into sleeping. Sleeping nodes wake up once in a while to probe their neighborhoods and replace any failed working nodes as needed. To be robust and implementable on small sensors with stringent resources, 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 estimations of their neighbors; they achieve distributed coordination by

*A Shorter version of this paper appeared in ICDCS 2003.

F. Ye (✉)
IBM T.J. Watson Research,
19 Skyline Drive, Hawthorne, NY, 10572
e-mail: fanye@us.ibm.com

S. Lu · L. Zhang
Computer Science Department, UCLA,
Los Angeles, CA 90095-1596
e-mail: {slu; lixia}@cs.ucla.edu

H. Zhang · J. Hou
Computer Science Department, UIUC,
Urbana, IL, 61801-2302
e-mail: hzhang3@uiuc.edu
e-mail: jhou@cs.uiuc.edu

a simple probing mechanism. A node only needs to find out whether any working neighbor exists within a local probing range to decide whether it should be turned off or not. The wakeup frequencies of sleeping nodes are self-adjusted to both maintain adequate working node density and minimize energy consumption. As shown by our analysis and simulation results, PEAS ensures asymptotic network connectivity; it can extend a sensor network's functioning time in linear proportional to the number of deployed nodes, using less than 1% of the total energy consumption and withstanding up to 38% of node failures.

Different from the protocols designed for ad-hoc networks that assume dynamic changes in connectivity but no frequent node failures, PEAS targets at a harsh working environment in which node failures happen frequently. PEAS design also differs from existing energy-saving protocols that exploit node redundancy, such as GAF [3], SPAN [4], ASCENT [5], and AFECA [6]. The fore-mentioned protocols are targeted for either ad-hoc networks or a relatively stable sensor network environment where nodes do not fail unexpectedly. Although they all maintain a stable number of working nodes in the presence of battery depletions, their operations either depend on the predictability of individual nodes' lifetimes, or require each node maintain state for every neighbor. In contrast, PEAS assumes that the density of deployed nodes may be orders of magnitude higher than that of the working ones, 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 its connectivity, complexity in Section 3. We address several practical implementation issues in Section 4, and present the performance evaluation of PEAS and compare it with GAF [3] 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 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 [7, 8].

2. PEAS design

2.1. Overview

PEAS works for ad hoc sensor networks that consist of large numbers of densely-deployed, small, inexpensive sensors. Such networks find applications in forest fire monitoring, military surveillance, ocean environmental sampling, etc. Due to adverse external factors, these nodes may suffer frequent and unpredictable failures. The network relies on a certain degree of redundancy in working nodes for robust functioning. Nodes are usually deployed at much higher densities than the minimum required for extended system lifetime. We assume that each sensor node can vary its transmission power to choose a power level to cover a circular area given a radius.[1]

The goal of PEAS is to extend system lifetime by exploiting the high redundancy in deployment, while being robust against severe, unpredictable node failures. The basic approach is to turn off redundant nodes. Two main issues are how to decide which nodes to turn off and how long they sleep. The two components of PEAS, Probing Environment and Adaptive Sleeping, address these issues, respectively.

To decide which sensors to turn off, nodes need to coordinate among themselves. This is typically achieved based on each node's knowledge about its neighborhood, such as each neighbor's location, connectivity, etc. However, in a dense network with frequent, unpredictable node failures, keeping track of each neighbor's status can be very difficult; the constrained memory and energy resources of each node further aggravate the situation. We thus strive to avoid per neighbor state, eliminating the burden of maintaining such state. The real challenge is, without the knowledge of each neighbor, how do nodes achieve distributed coordination? This is what Probing Environment answers. It utilizes a probing mechanism by which a node discovers whether a working one exists in a certain probing range and decides whether it should be turned off. It determines the topology of working nodes while requiring minimum amount of knowledge of local neighborhood.

The next observation is that, how long nodes sleep decides how quick a dead working node can be replaced because only when sleeping nodes wake up can they replace dead working neighbors. Each unexpected death of working nodes causes an interruption in sensing and communication. The lengths of such interruptions should be kept within what are tolerable from the application's perspective. Adaptive Sleeping decides when a sleeping node should wake up. It keeps the aggregate wake up rate of the sleeping neighbors of each working node at appropriate levels desired by the application, in order to meet application requirements and minimize the control overhead.

2.2. Probing environment

We describe Probing Environment in this section. Each node in PEAS has three operation modes: *Sleeping, Probing* and *Working*. The state transition diagram among these three modes is shown in Fig. 1. After nodes are deployed, they

[1] Existing hardware e.g., MOTES, already allows variable transmission power [1]. We discuss how PEAS works with fixed transmission power in Section 4.

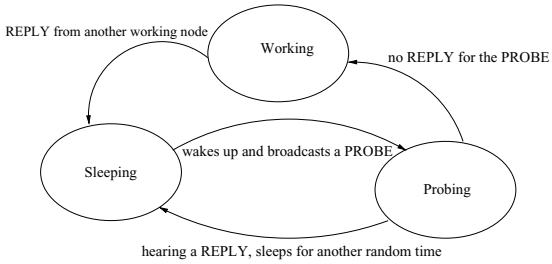


Fig. 1 State transition diagram of operations at each node

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 time 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. λ is adjusted according to the Adaptive Sleeping algorithm in Section 2.3 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.

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.

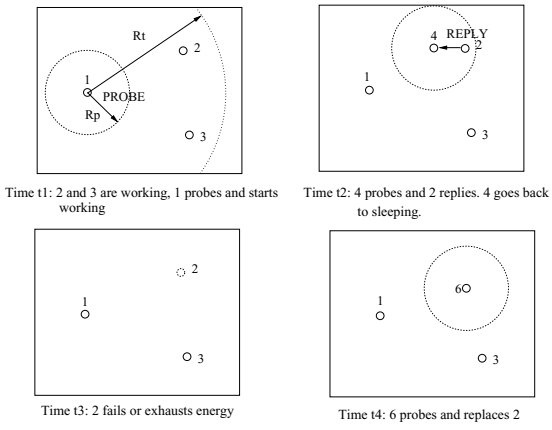


Fig. 2 An example of probing environment

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. Through the above example, we can see the result of this probing is to keep a distance of at least R_p between any two adjacent working neighbors. Thus nodes can only work when they are far enough from existing working ones.

It is possible that PROBE and REPLY messages are still lost due to factors such as collisions or link errors. In such cases, a probing node supposed to sleep again will start working, unnecessarily wasting energy. To correct such errors, working node react to REPLYs sent by its working neighbors that 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. One of them should be turned off. In practice, if either of the two can turn off the other, they may take turns to work and cause an unstable working node topology. Many routing protocols have to rebuild routing state in new working nodes and suffer from unstable topologies. We favor the one that has been working for a longer time to stabilize the topology: Each working node records the time when it started 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.

The above design has some parameters that need to be set. The initial value of λ decides how quickly the network acquires enough number of working nodes during the boot-up phase and how fast dead working nodes are replaced. For instance, the application requires the network start functioning 1-minute after deployment and 50% of the deployed nodes are needed. 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 of working nodes. It is specified by the application based on its requirements for both robust sensing and robust communication. These two functions may require different densities of working nodes. For example, a type of sensors can detect animals within 10 m and transmit up to 20 m. Suppose the application decides that working nodes should be spaced at most 3 m for robust sensing, but 6 m are enough for robust communication. The application may simply choose the probing range R_p as the smaller value of 3 m[2]. The choice

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

of R_p also affects network connectivity; it must be less than the maximum transmitting range R_t to avoid disconnection. This is to be analyzed in Section 3.

2.2.1. *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 rule. Location-based probing rule ensures that adjacent working nodes keep appropriate distances, which translate into desired redundancy for resilient sensing and communicating functions. This feature is important because excessively 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 for robustness, e.g., frequent node failures cause part of the field unmonitored.

Unlike related schemes for conventional ad hoc networks that choose to turn on nodes with more energy or more neighbors [3, 4], PEAS does not favor such nodes. This is motivated by the characteristics of the kind of sensor networks it targets. The system relies on the *collective* behavior of nodes to fulfill the same task reliably. As long as PEAS maintains enough working nodes, they can perform required sensing and communication tasks, thus the capability of individual node does not matter. However, in conventional ad hoc networks, each node may belong to a different user, thus attention must be paid not to exhaust the energy of individual nodes.

We want to point out that there is a tradeoff between robustness and optimality. In the design, we intentionally make choices that lead to simplicity, thus better robustness. Although it is possible to turn off redundant nodes and achieve certain optimality utilizing per neighbor state about local topology, PEAS avoids such state. It achieves *distributed* coordination by keeping distance between working nodes. This simple design enables PEAS to adapt to high deployment densities and to be implementable on very small nodes. Further analysis about the effectiveness of probing will be given in Section 3.

Randomized sleeping time. A node in PEAS sleeps for a randomized period of time. The wakeups of nodes are spread over time (shown in Fig. 3). This is different from some related schemes [4] that 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 (shown in Fig. 4).

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 anticipated lifespan, there come large "gaps' during which no working node is available (illustrated in Fig. 5).

Therefore, PEAS chooses to distribute node wakeups over time by randomization, rather than to cluster them at synchronized time points. Shown in Fig. 6, node wakeups come at much shorter time intervals. Thus the average gap between two successive working nodes in any local neighborhood can be greatly shortened. The frequency at which the sleeping neighbors wake up and probe decides how quick a dead working node is replaced. This frequency is adjusted by Adaptive Sleeping to make the "gaps" tolerable by applications.

A remaining question about Probing Environment is why it uses exponential distribution for the random sleeping time. We will show in Section 2.3 that exponential distribution leads to a Poisson process of probing events; this exhibits nice properties that the Adaptive Sleeping exploits. An analysis of the gap length will be given in Section 3.3.

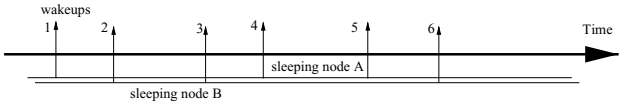


Fig. 3 In PEAS, sleeping nodes A and B have randomized sleeping times

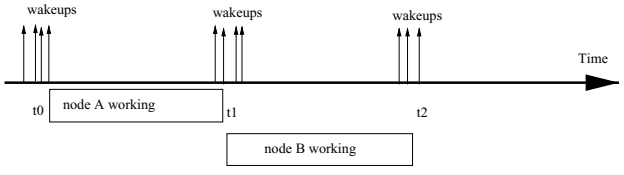


Fig. 4 Deterministically synchronized wakeups

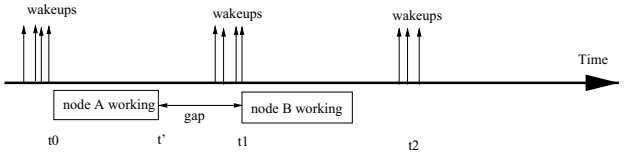


Fig. 5 Synchronized operation has big gaps when nodes fail

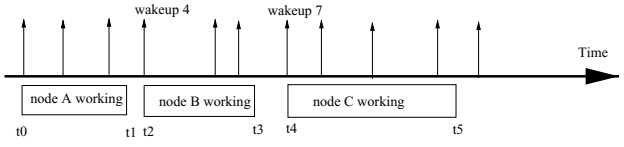


Fig. 6 Distributing wakeups over time shortens gaps

2.3. Adaptive sleeping

Adaptive Sleeping adjusts the probing rate λ of each sleeping node so as 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 caused by node deaths are acceptable to the application, while the probing overhead is minimized.

The design issue is that, the number of sleeping neighbors of a working node changes over time and varies in different locations. Thus a fixed per node λ cannot ensure a desired aggregate probing rate. Each node has to adjust its λ 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 pieces of state:

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

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 increases by one. Eventually when the counter reaches a threshold value k, a measurement $\hat{\lambda}$ of the actual probing rate $\bar{\lambda}$ is calculated as follows: [3]

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

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 7 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 $\hat{\lambda}$:

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

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

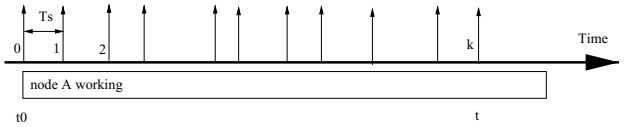


Fig. 7 Measure $\bar{\lambda}$

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

2.3.1. Explanation

We now explain why the above algorithm keeps the aggregate $\bar{\lambda}$ around the desired λ_d. From the probability theory [9], 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:

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

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 estimation $\hat{\lambda}$ 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 [9] 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.

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. Based on the same principle of trading optimality for simplicity and better robustness, we opt for the simplest possible design to make PEAS resilient in a dynamic environment.

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. Performance analysis of PEAS

The performance analysis includes several parts. The connectivity analysis derives conditions under which the working nodes are a.a.s. (asymptotically almost surely) connected. This is important to ensure that the probing rule does not cause disconnected networks. Next we give the density bounds of working nodes, and the average time needed to replace nodes that fail unexpectedly. The complexity analysis characterizes the state, energy and computing overheads of PEAS.

3.1. Asymptotic connectivity of PEAS

We first present the following PEAS model to aid the analysis. Consider a two-dimensional network field.[4] 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[5] is at least $R_p/2 + R_p/2 = R_p$, which is 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. Thus two peas can replace these two working nodes without overlapping. So any possible positions of working nodes is *equivalent* to a placement of peas on the plane.

To find out under what conditions 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 give the conditions under which each cell has at least one node a.a.s. based on Blough's Theorem 2 in [10], then we prove that given each cell has at least one node, PEAS ensures connectivity a.a.s. when the transmitting range $R_t \geq (1 + \sqrt{5})R_p$.

The following Lemma 3.1. tells us under what conditions each cell has at least one node a.a.s. The proof is given in the Appendix.

Lemma 3.1. *When n nodes are uniformly randomly distributed in* $R = [0, l]^d$, *for* $d = 2$, *and assume that* $c^d n = kl^d \ln l$ *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 conditions in Lemma 3.1 are satisfied, i.e., each cell has at least one node a.a.s., for any 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 loss of generality, we let node A center at the middle of cells 1, 2, 3 and 4 (Fig. 8). 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 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, thus are all sleeping. Consider node C in cell 5, which is to the right of cell 1. Given that node C is randomly 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 sleep, another node B must be working within the probing range of C. Based on geometry calculations, the farthest 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

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

[5] We use "PEAS' for the protocol and "peas' for the seeds of the plant.

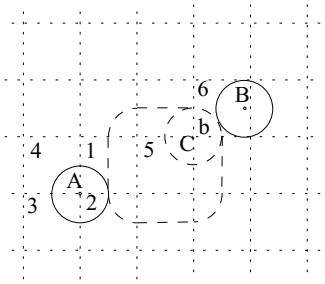


Fig. 8 The minimum distance between two adjacent working nodes

always be working, no matter where it is located within cell 5.

Now 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 $O(l^2)$, $P(\min(\text{Dist}(A, B)) < (1 + \sqrt{5})c)$ still approaches 1 as $l \to \infty$.

The following theorem stipulates the condition for asymptotic connectivity in PEAS. It shows that the probing range has to be set properly to ensure asymptotic network connectivity.

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 it 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 (Fig. 9). Any pea (working node) in S_1 is to the left of L. Following similar reasonings in Lemma 3.1, 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 is contradictory to the assumption that any working node in S_1 is connected to only others in S_1. Thus there is no such disconnected component. Note that based on similar reasoning, the boundary cases does not affect the asymptotic connectivity as $l \to \infty$.

3.2. Densities of working nodes

Here we derive the maximum and minimum densities of working nodes under the assumption that the deployment density is very high. The upper bound is due to the probing rule that any two working nodes keep a distance of at least R_p. With the PEAS model, the maximum density corresponds to

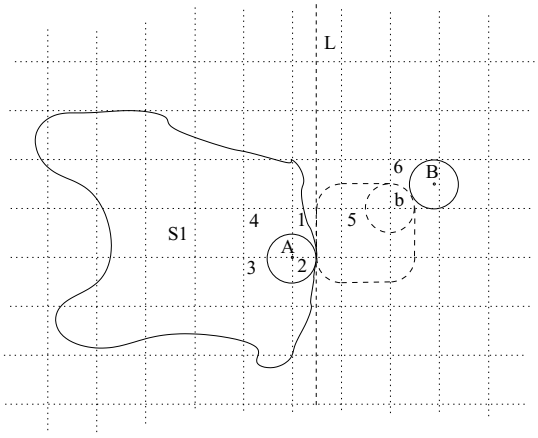


Fig. 9 A "disconnected' component always has another working node connected

the most compact case of peas placement in a two dimensional plane. It happens when each pea is tangent with six neighboring peas (as shown in Fig. 10). Using geometry calculations, we derive the maximum working node density as

$$\rho_{\max} = \frac{2}{\sqrt{3}R_p^2}. \tag{4}$$

On the other hand, given enough deployed nodes, the minimum node density is also lower bounded (see Fig. 11). This is because when the centers of two peas are far away, there is enough space to insert another pea in between. In the minimum density case (shown in Fig. 11), each pea still has 6 neighboring peas, but the space among any three adjacent nodes is slightly smaller than a pea—thus another pea cannot be inserted in between. The minimum working node density can be calculated as

$$\rho_{\min} = \frac{2}{3\sqrt{3}R_p^2}. \tag{5}$$

These two theoretical bounds, however, have little chance to appear in reality because they require precise hexagonal

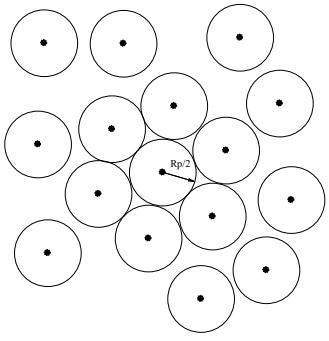


Fig. 10 7 tangent peas in the center is the most compact case

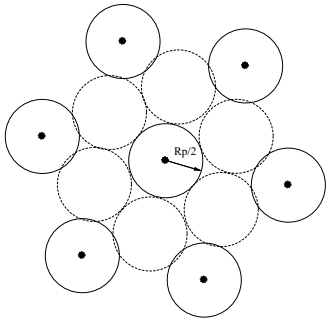


Fig. 11 Lower bounding the density given enough deployed nodes

placements of nodes. The randomized probings in PEAS make such accurate placement almost impossible. Experiments of uniformly randomly deployed nodes show that the actual density distribution falls into a narrower range, with the ratio of the minimum density to the maximum density being 1:1.14. This indicates that, given a dense deployment, the density of working nodes can be kept roughly constant at different places (The probability density distribution of actual density is shown in Figure 12).

3.3. Average gap $\bar{g}$ between successive working nodes

A gap is the period from the death of a working node (due to failures or energy depletion) to the time when a probing node replaces it. Its length decides how long local sensing and communications are interrupted or degraded due to lack of a working node. It should be within what is tolerated by the application. Due to the memoryless property of exponentially distributed sleeping times, the average gap length, which is from the death of a working node to next wakeup, should be the same as the average interval between wakeups. That is,

$$\bar{g} = \frac{1}{\bar{\lambda}}. \tag{6}$$

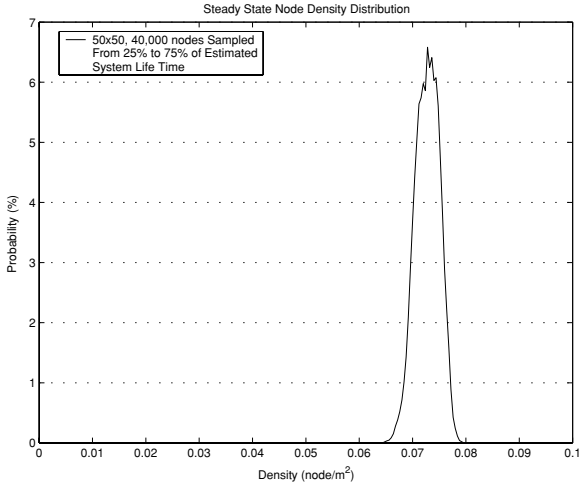


Fig. 12 Probability density function of actual working node density

Experiments also confirmed the above equation. The application may use the above to decide λ_d on how frequent sleeping nodes should wake up, based on its tolerable $\bar{g}$ value.

3.4. State, computing and energy complexities

The state overhead at any node is $O(1)$. Each working node maintains a constant amount of state, including counter N, last counter reset time t_0, the measured probing rate $\hat{\lambda}$ and the desired probing rate λ_d. Each sleeping node maintains state of its probing rate λ and probing range R_p. There is no per-neighbor state maintained at each node. The state at any node is independent of deployment density and network size.

The computing and energy overheads are in proportional to the number of wakeups. Each wakeup consumes energy for transmitting PROBEs and REPLYs, and waiting for REPLYs. Computation is incurred when working node updates its measurement or a probing node adjusts its probing rate.

Because of Adaptive Sleeping, the number of wakeups in a local neighborhood per unit time is adjusted to be approximately constant, independent of the deployment density. Therefore, the computing and energy overheads in each local neighborhood per unit time are both $O(1)$.

4. Discussions

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 to minimize overhead.

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

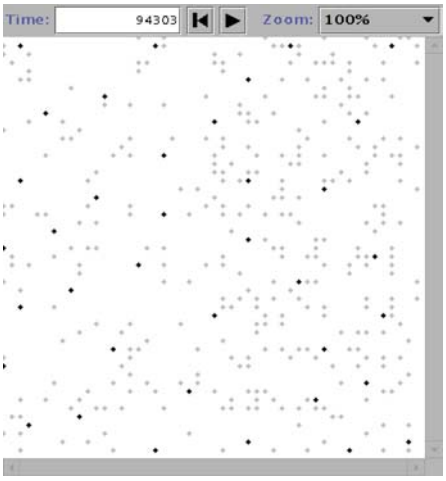


Fig. 13 Black and gray dots are working and sleeping nodes, respectively. The right half has twice as many deployed nodes as the left half, but they both have about 18 working nodes

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 (See Fig. 13 for an example). 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 conjecture that evenly deployed nodes will work longer than those deployed irregularly.

5. Performance evaluation

5.1. Methodology and metrics

We implement PEAS in PARSEC [11] and select sensor hardware parameters similar to Berkeley Motes [1]. The node power consumptions in transmission, reception, idle and sleep modes are 60 mW, 12 mW, 12 mW and 0.03 mW, respectively. The initial energy of a node is randomly chosen from the range of 54–60 Jules to simulate the variance of battery lifetime, allowing the node to operate about 4500 – 5000 seconds in reception/idle modes.[6] The sensing and maximum transmitting ranges are both 10 m. Each node has a raw wireless communication capacity of 20 Kbps. The packet size of PROBE and REPLY messages is 25 bytes, which is enough to hold the information they need to carry.

We use a 50×50 m^2 network field, and nodes are uniformly distributed in the field initially and remain stationary once deployed. A source and a sink sit 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 [12]. 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 m. 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 uniformly randomly distributed over time in the simulation. The *failure rate* denotes the average number of failures per unit time. The *failure percentage* is the percentage of failed nodes. Note that failures are sensor deaths incurred by factors other than energy exhaustion.

The main metrics used are *sensing coverage lifetime* and *data delivery lifetime*. The *sensing coverage* is defined as the percentage of of the field size 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. K-coverage's lifetime is the time duration from the beginning until K-coverage drops below a threshold value. It depicts how long the system can ensure that the occurrence of any interested events can be 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 describes for how long the network can deliver reports to users. Both threshold values are chosen as 90%. In addition, we measure the overhead of PEAS by the number of wakeups and calculate the energy consumed by its operations.

To understand how PEAS performs compared to related work, we also measure the sensing coverage and data delivery lifetimes of a revised version of GAF [3]. We choose GAF because it can easily be adapted to maintain sensing coverage: given the sensing range, GAF can ensure each point be monitored by at least K nodes by letting each grid have K working nodes. We do not select SPAN [4] because it was designed for mobile wireless networks only; it maintains connectivity but not sensing coverage. We find that PEAS outperforms the revised GAF in extending system lifetime. More details will be presented in Section 5.4.

[6] This is much lower than the Motes hardware can provide (about 200 hours in idle) so simulations do not take weeks to finish. It does not change the conclusions from the results.

5.2. Prolong system functioning time

To see how PEAS adapts to varying node population, we set the node number as 160, 320, 480, 640 and 800 and simulate for long enough 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 m, 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. For each node population, we simulate 5 different runs and average the results. All runs in this section use 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. Figure 14 shows the lifetimes of 3, 4 and 5-coverage. As the deployment population increases, each lifetime increases almost linearly. This is because PEAS keeps only a necessary number of nodes in working, while turning off others to sleep. The more deployed nodes, the more in sleeping, and the longer they can keep the sensing coverage. We also see that the lifetimes of 3-coverage are longer than corresponding ones of 4-coverage, because fewer working nodes are required to ensure each area being covered by at least 3 nodes. This is also the case for 4- and 5-coverage.

The data delivery lifetime is shown in Fig. 15. 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 decrease to the 90% threshold.

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.

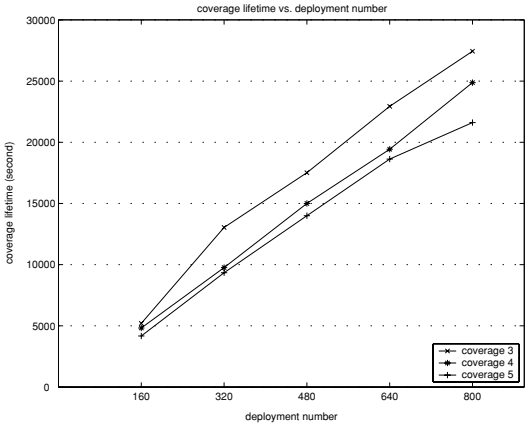


Fig. 14 Extension of coverage lifetime

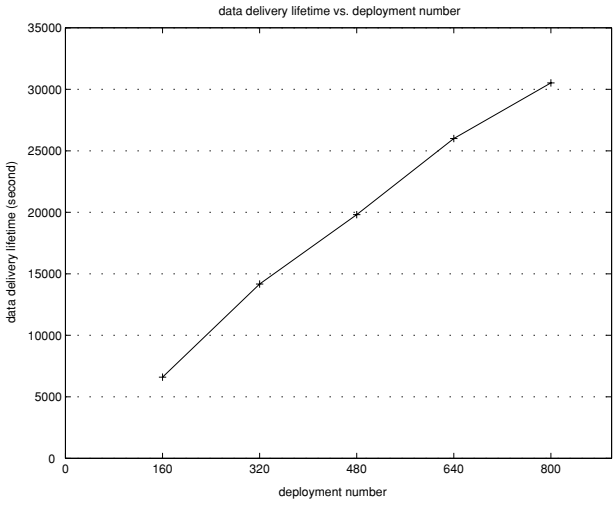


Fig. 15 Extension of data deliveryl lifetime

We then look at the overhead incurred for PEAS' operations. Figure 16 shows the average number of wakeups for each deployment number. This number also increases linearly as the node population increases. This is because Adaptive Sleeping adjusts the wakeup frequency to the desired level. When the network functions longer, proportionally more wakeups happen.

To measure the energy overhead, we first calculate the energy consumed in each wakeup. The energy used in a wakeup consists of that in transmitting and receiving PROBE and REPLY messages, and in the time a probing node waits in idling to receive REPLYs. Based on the current implementation in which a probing node transmits three PROBEs and wait 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 to the total energy consumption in

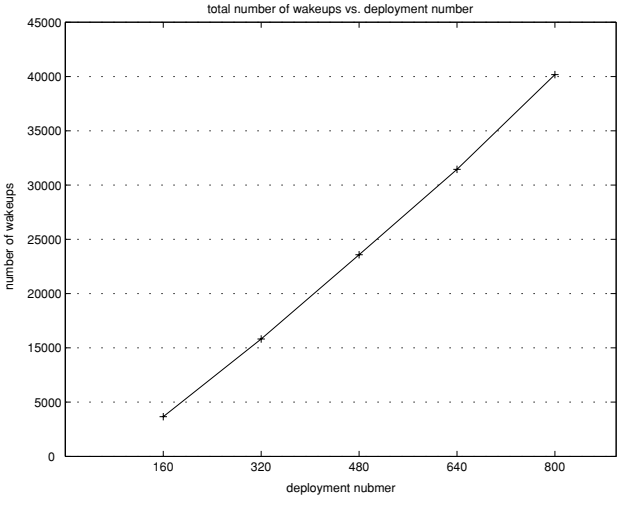


Fig. 16 Average total wakeup count for deployment numbers

Table 1 Energy overhead for deployment numbers

Node number	160	320	480	640	800
Energy overhead (J)	11.58	34.18	58.68	83.53	111.11
Total energy (J)	8082.15	16498.07	24872.13	33382.86	41680.49
Overhead ratio (%)	0.143	0.207	0.236	0.25	0.267

Table 2 Failure percentage

Failure rate	5.33	10.66	16	21.33	26.66	32	37.33	42.66	48
Failure percentage (%)	7.083	12.92	17.46	21.42	25.83	28.62	31.75	34.96	38.54

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 8 increments of 5.33 each. For each failure rate, we show the average failure percentage—the ratio of failed nodes to the total number deployed in Table 2. 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 normally at all.

Figure 17 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 nodes in the sleeping mode to combat against 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 some random factors).

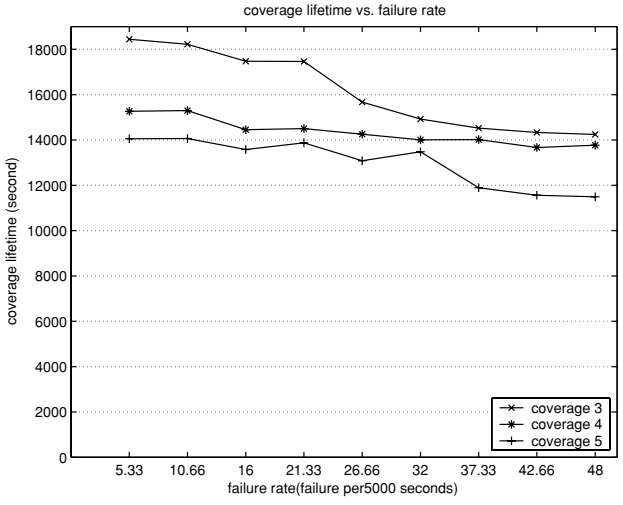


Fig. 17 Coverage lifetime with failures

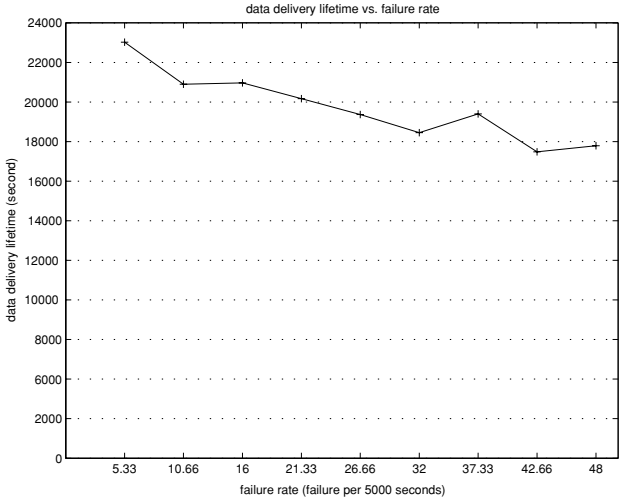


Fig. 18 Data delivery life time with failures

The average data delivery lifetime for each failure rate is shown in Fig. 18. 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 (Fig. 19). 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 rather than consuming more energy to fight against failures, PEAS achieves robustness at roughly constant overhead.

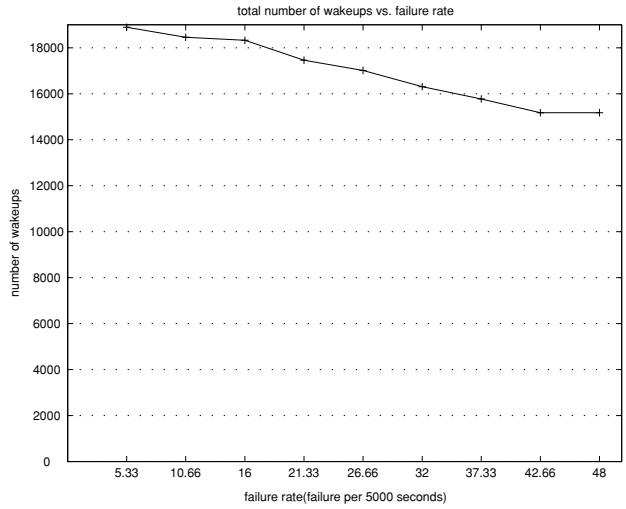


Fig. 19 Average total wakeup count for failure rates

5.4. Comparison with GAF

To see how well PEAS performs compared to existing work, we compare the sensing coverage and data delivery lifetimes of PEAS to those of a revised GAF [3]. We modify GAF as follows to maintain sensing coverage. First, using hexagon cells are more efficient in reducing the coverage overlap, thus maintaining the same coverage with less working nodes. So we divide the whole area into hexagon cells rather than square grids in [3]. Second, we let the side length of each hexagon cell be half of the sensing range, so any node in a hexagon cell can completely cover the cell. Third, in order that each point is covered by at least K working nodes, the revised GAF tries to maintain K working nodes in each cell. We choose AODV, the same routing protocol used in [3], to measure the data delivery lifetime of GAF. We choose the same system parameters (such as the node power consumptions, initial energy, etc) as in Section 5.1 in order that both protocols are subject to the same scenarios. Similar to Section 5.2, we assume the application requires that each point be covered by at least 4 working nodes, i.e, the revised GAF tries to maintain 4 working nodes in each cell. We choose the same definitions of the coverage lifetime and data delivery lifetime as those in Section 5.1 and 5.2.

First we compare how they perform when the number of deployed nodes increases. Figure 20 shows the 3, 4 and 5-coverage lifetimes of the revised GAF. We find that similar to PEAS, GAF extends coverage lifetime linearly as more nodes are deployed. However, the lifetimes are 10% to 20% lower than corresponding ones of PEAS (Fig. 14). The data delivery lifetime of GAF is shown in Fig. 21. Although GAF increases the lifetime as more nodes are deployed, data delivery stops earlier than that of PEAS (Fig. 15). On average, GAF achieves 70–80% of the data delivery lifetime of PEAS. For example, with 480 nodes PEAS has a data

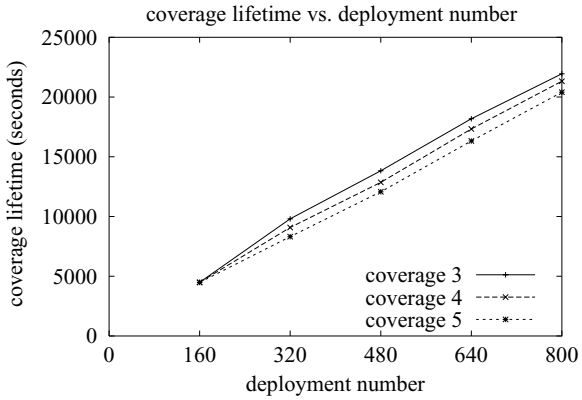


Fig. 20 Extension of coverage lifetime using GAF

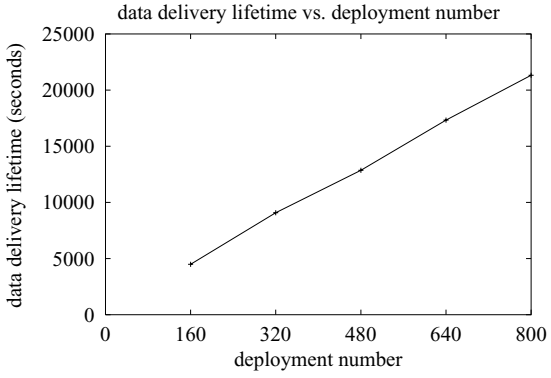


Fig. 21 Extension of data delivery lifetime using GAF

delivery lifetime about 20000 seconds, whereas GAF about 13500 seconds.

To understand why PEAS outperforms GAF, we investigate how the number of working nodes changes over time for both protocols. We find that to maintain the same degree of sensing coverage, GAF tends to keep more nodes working than PEAS. For example, with 640 deployed nodes, GAF has about 156 working nodes, but PEAS has only about 140.

There are two reasons why PEAS can use less nodes to maintain the same coverage. First, in GAF, working nodes in adjacent cells can be very close to each other, thus their sensing areas have much overlapping. Such topologies are allowed as long as neighboring nodes belong to different cells. However, in PEAS, two working nodes have to be R_p apart from each other, which effectively prevents excessive overlapping of the sensing areas. Second, the *fixed* and *predefined* regular cell structure of GAF can not adapt to the actual node deployment topology in every location. Given the cell size, the cell boundary division (where the cell boundaries are drawn) decides which nodes belong to different cells, thus can work simultaneously. However, the cell boundary division that minimizes the working node number at one location may not coincide with the division at another location. Thus it is impossible to minimize the working node number everywhere at the same time. In PEAS, due to the

Table 3 Total data packets delivered

Failure percentage (%)	0	13	27	40	53
PEAS	276	233	202	162	158
GAF	122	121	111	107	98

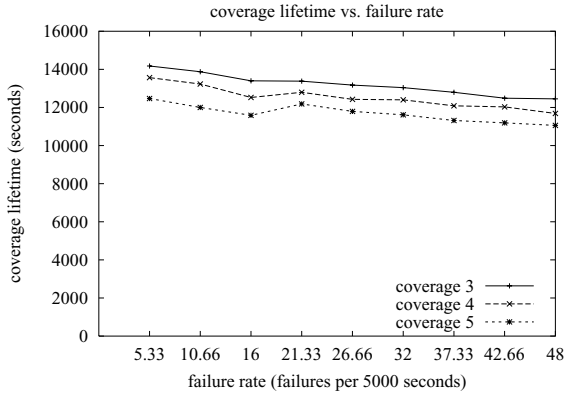


Fig. 22 Coverage lifetime with failures using GAF

lack of such constraints imposed by a predefined structure, the topology of working nodes at different locations can be optimized simultaneously.

Because less working nodes are needed, PEAS keeps more nodes in sleeping and conserves more energy. Eventually, when GAF exhausts working nodes, PEAS still have enough for sensing and delivery. This is why PEAS can achieve longer coverage and data delivery lifetimes. This is also confirmed by the total number of packets successfully delivered by both protocols. Table 3 shows the comparison for 150 deployed nodes with different node failure percentages. We can see that PEAS supports the delivery of at least 50% more packets than GAF does.

Now we compare the robustness of PEAS with GAF. We choose the same scenario as in Section 5.3. Figures 22 and 23 show how the coverage and data delivery lifetimes of GAF change as more nodes fail. We make similar observations: (1) the lifetimes decreases as more nodes fail; (2) the lifetimes are 10–30% lower than those of PEAS (as in Figs. 17 and 18). The result is consistent with the previous comparison and confirms that PEAS possess better robustness under significant node failures.

6. Related work

To preserve the limited battery power, various approaches have been explored to place unnecessary nodes into the sleeping mode for both wireless ad hoc networks (e.g. GAF [3], SPAN [4], AFECA[6]) and sensor networks (e.g. [5, 13–16]).

In previous work, SPAN maintains a list of working neighbors at each node and exchanges the lists among neighbors. All nodes utilize this 2-hop neighborhood topology to turn

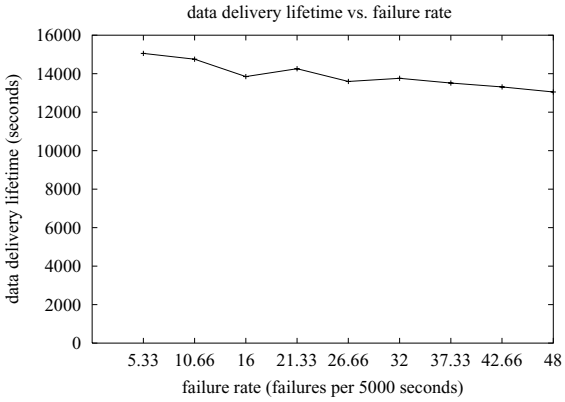


Fig. 23 Data delivery lifetime with failures using GAF

on nodes that connect more neighbors or have more energy. All sleeping nodes wake up periodically to re-elect working ones. GAF assumes that each node knows its location through GPS or other location services. It divides the network into geographical square grids (called cells). Within each cell, a node is active while the rest go to sleep, with the sleep time being set based on the remaining energy of the working one. In AFECA, each node keeps a list of neighbors to track the number of neighbors. A sleeping node dozes for a random period of time in proportional to the number of neighbors. ASCENT [5] proactively measures the number of active neighbors and per-link data loss rates. Each node decides whether it should work or sleep based on a number of rules regarding the above measurement.

A number of proposals have emerged following the sensing coverage study in our earlier work [12], where PEAS is one component of an integrated system. In [13], each node calculates the "sponsored area' (defined as maximum sector covered by a neighbor) provided by each of its neighbor. A node goes to sleep if the union of all its neighbors' sponsored area completely covers its coverage disk. Gupta et al. [14] devise both a centralized and a distributed algorithm to find a subset of nodes that ensure both coverage and connectivity. The centralized algorithm guarantees that the size of the formed subset is within $O(\log n)$ factor of the optimal size, where n is the network size. Both [15] and [16] notice that if transmission range is at least twice of sensing range, coverage implies connectivity, and if every crossing point (intersection point of coverage disks or that of coverage disks and monitored region) in a region is covered, the whole region is completely covered. In [15] each node uses this as a rule to determine if its coverage disk is completely covered and if it should go to sleep. Zhang *et al.* [16] further investigate what is the optimal location of a node to cover a crossing point and then choose a node that is closest to the optimal location to cover an uncovered crossing point.

To prolong the system lifetime, PEAS uses the same basic principle of turning off unused nodes to preserve energy. However, in order to be both resilient against unpredictable node failures in a harsh or even hostile environment, and versatile under various degrees of deployment density, nodes in PEAS do *not* keep any per-neighbor information. PEAS utilizes a randomized 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 or not. This probing also controls the degree of overlapping of the sensing areas of neighboring working nodes. It prevents excessively dense working nodes and adapts to local deployment topologies, as demonstrated in the quantitative comparison in Section 5.4. Instead of relying on predefined virtual grid (GAF), or information difficult to predict in a harsh environment (such as the lifespan of a working node in SPAN), PEAS design exploits randomization and adaptivity to achieve simplicity and resiliency.

STEM [17] and S-MAC [18] exploit a different dimension to conserve energy. They control the duty cycles of nodes to reduce the energy consumed in idle listening. They are suitable for sensor networks where data traffic happen infrequently, thus most of the time the radios of nodes can be turned off. These energy saving techniques in temporal dimension are orthogonal to the ones in spatial dimension of PEAS. They can be combined to achieve further energy efficiency, as demonstrated in [17].

Energy can also be conserved through the MAC layer. PAMAS [19] turns off the radio of a node when the traffic is not addressed to it. TDMA is proposed to reduce the duty cycle of nodes to save energy [20]. They do not address the robustness and scalability issues PEAS considers.

There are different types of sensor hardware. WINS [21] are powerful ones similar to mobile computers in wireless ad hoc networks. Medium-size ones include Rockwell nodes [22]. Other nodes have very low cost and severely constrained resources, including Berkeley Motes [1], smart dust [23], piconet nodes [24]. PEAS has overhead and complexities within the capabilities of all these hardware. It is designed to run on even the low-end ones with very limited resources.

7. Conclusion

It is becoming economically feasible to build large-scale sensor networks with large quantities of inexpensive and simple sensor nodes. In principle, the sheer scale of such sensor networks may overcome the limitations of individual nodes by exploiting the collective behavior of sensors. However, how to build a resilient, long-lived network with unreliable, short-lived sensors remains a challenge. The real operating environment can be harsh or even hostile; node failures can become norms rather than exceptions. Existing energy-saving protocols have not paid adequate attention to address unexpected node failures in a sensor network built on a large number of simple and fallible sensors.

This paper presents PEAS, a distributed and randomized energy-saving protocol for sensor networks. The fully randomized and asynchronous operations of PEAS ensure robustness in the presence of unexpected failures. The simplified protocol operations and minimized maintained state enhance robustness and make PEAS implementable on low-end nodes. PEAS can keep the working node density approximately constant independent of the total number of deployed nodes, thus prolonging system lifetime proportionally. Our simulations and analysis have confirmed the effectiveness of the design.

8. Appendix

The proof is similar to that of Theorem 2 in [10]. The number of cells $N = l^d/c^d$. Since each node is independently, randomly placed into one of the cells, which has been studied in occupancy theory [25]. The expected number of empty cells [25]

$$E[\mu_0(n)] = N\left(1 - \frac{1}{N}\right)^n$$

If each cell has at least one node, the number of empty cells is 0. To make the above goes to 0 as $l \to \infty$, we have

$$\begin{aligned}\ln E[\mu_0(n)] &= \ln N + n \ln\left(1 - \frac{1}{N}\right) \\ &= d \ln \frac{l}{c} + n \ln\left(1 - \frac{c^d}{l^d}\right)\end{aligned}$$

Since $c/l \to 0$ as $l \to \infty$, the last term can be approximated by its Taylor expansion,

$$\ln E[\mu_0(n)] \approx d \ln \frac{l}{c} - n\frac{c^d}{l^d}$$

Given $nc^d = kl^d \ln l$, ln the above becomes

$$d \ln \frac{l}{c} - k \ln l = \ln \frac{1}{c^d l^{k-d}}$$

If $k > d$, then $\lim_{l\to\infty} \ln E[\mu_0(n)] \to -\infty$, then $\lim_{l\to\infty} E[\mu_0(n)] = 0$. This implies that each cell has at least one node a.a.s., otherwise the expected number of empty cells

would not approach 0. Note that in the above proof d can take values of 1, 2 and 3, and Lemma 3.1 holds for one and three dimension cases also.

References

[1] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler and K. Pister, system architecture directions for networked sensors, in: *International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-IX)*, (2000).

[2] Single chip sensor nodes," http://www.cs.berkeley.edu/∼jhill/spec/,(2003).

[3] Y. Xu, J. Heidemann and D. Estrin, Geography informed energy conmservation for ad hoc Routing, in: *ACM International Conference on Mobile Computing and Networking (MOBICOM' 01)*, (2001).

[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] 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 2002).

[6] Y. Xu, J. Heidemann and D. Destrin, Adaptive energy-conserving routing for multihop ad hoc networks,USC/ISI Research Report 527, (Oct. 2000).

[7] F. Ye, G. Zhong, S. Lu and L. Zhang, A Robust data delivery protocol for large scale sensor networks, in: *IPSN* (2003).

[8] C. Intanagonwiwat, R. Govindan and D. Estrin, Directed diffusion: A scalable and robust communication paradigm for sensor networks, in: *ACM International Conference on Mobile Computing and Networking (MOBICOM'00)* (2000).

[9] S. Ross, Introduction to Probability Models (6th ed.) (Acedemic Press, 1997).

[10] 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).

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

[12] F. Ye, S. Lu and L. Zhang, "GRAdient Broadcast: A robust, long-lived large sensor network," http://irl.cs.ucla.edu/papers/grab-tech-report.ps (2001).

[13] D. Tian and N.D. Georganas, A coverage-preserving node scheduling scheme for large wireless sensor networks, in: *First ACM International Workshop on Wireless Sensor Networks and Applications*, Georgia, GA (2002).

[14] H. Gupta, S. Das and Q. Gu, "Connected sensor cover: self-organization of sensor networks for efficient query execution," in *Proc. of Mobihoc* (2003).

[15] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless and C. Gill, Integrated coverage and connectivity configuration in wireless sensor networks, in: *ACM Sensys'03* (Nov. 2003).

[16] H. Zhang and J.C. Hou, "Maintaining sensing coverage and connectivity in large sensor networks, in: *International Workshop on Theoretical and Algorithmic Aspects of Sensor, ad hoc Wireless and Peer-to-Peer networks* (Feb. 2004), also a technical report with reference number UIUCDCS-R-2003-2351 in the department of computer science, university of Illinois at Urbana-Champaign.

[17] C. Schurgers, V. Tsiatsis, S. Ganeriwal and M. B. Srivastava, Optimizing sensor networks in the energy-latency-density design space, *IEEE Transactions on Mobile Computing*, 1 (1) (2002).

[18] W. Ye, J. Heidemann and D. Estrin, An energy-efficient MAC protocol for wireless sensor Networks, in: *proceedings of the 21st International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2002), New York, NY, USA* (June, 2002).

[19] S. Singh and C. Raghavendra, PAMAS: Power aware multi-access protocol with signalling for ad hoc networks, ACM Computer Communication Review, 28 (3) 5–26 (July 1998).

[20] K. Sohrabi and G. Pottie, Performance of a novel self-organization protocol for wireless ad hoc sensor networks, in: *Proceedings of IEEE VTC, Amsterdam, Netherlands* (Sept. 1999).

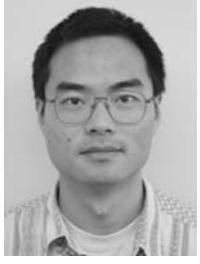
[21] G. Pottie and W. Kaiser, Wireless integrated network sensors, Communications of the ACM, 43 (5), 51–8, (May 2000).

[22] J. Agre, L. Clare, G. Pottie, and N. Romanov, Development platform for self-organizing wireless sensor networks, in: *Proc. SPIE, Unattended Ground Sensor Technologies and Applications, 3713, pp. 257–268*.

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

[24] F. Bennett, D. Clarke, J.B. Evans, A. Hopper, A. Jones and D. Leask, Piconet: Embedded Mobile networking, *IEEE Personal Communications Magazine, 4 (5) 8–15*. (Oct. 1997).

[25] V. Kolchin, B. Sevast'yanov and V. Chistyakov, *Random Allocations*. V.H. Winston and Sons, 1978.



Fan Ye received his B.E. in Automatic Control in 1996 and M.S. in Computer Science in 1999, both from Tsinghua University, Beijing, China. He received his Ph.D. in Computer Science in 2004 from UCLA. He is currently with IBM Research. His research interests are in wireless networks, sensor networks and security.

Honghai Zhang received his BS in Computer Science in 1998 from University of Science and Technology of China. He received his MS and Ph.D. in Computer Science from University of Illinois at Urbana-Champaign. He is currently with the Wireless Advanced Technology Lab of Lucent Technologies. His research interests are wireless networks, WiMAX, and VoIP over wireless networks.

Songwu Lu received both his M.S. and Ph.D. from University of Illinois at Urbana-Champaign. He is currently an associate professor at UCLA Computer Science. He received NSF CAREER award in 2001. His research interests include wireless networking, mobile computing, wireless security, and computer networks.

Lixia Zhang received her Ph.D in computer science from the Massachusetts Institute of Technology. She was a member of the research staff at the Xerox Palo Alto Research Center before joining the faculty of UCLA's Computer Science Department in 1995. In the past she has served on the Internet Architecture Board, Co-Chair of IEEE Communication Society Internet Technical Committee, the editorial board for the IEEE/ACM Transactions on Networking, and technical program committees for many networking-related conferences including SIGCOMM and INFOCOM. Zhang is currently serving as the vice chair of ACM SIGCOMM.

Jennifer C. Hou received the Ph.D. degree in Electrical Engineering and Computer Science from The University of Michigan, Ann Arbor in 1993 and is currently a professor in the Department of Computer Science at University of Illinois at Urbana Champaign (UIUC). Prior to joining UIUC, she has taught at Ohio State University and University of Wisconsin - Madison. Dr. Hou has worked in the the areas of network modeling and simualtion, wireless-enabled software infrastructure for assisted living, and capacity optimization in wireless networks. She was a recipient of an ACM Recognition of Service, a Cisco University Research Award, a Lumley Research Award from Ohio State University, and a NSF CAREER award.