Distributed Monitoring and Aggregation in Wireless Sensor Networks

Changlei Liu and Guohong Cao
Department of Computer Science & Engineering
The Pennsylvania State University
E-mail: {chaliu, gcao}@cse.psu.edu

Abstract— **Self-monitoring the sensor statuses such as liveness, node density and residue energy is critical for maintaining the normal operation of the sensor network. When building the monitoring architecture, most existing work focuses on minimizing the number of monitoring nodes. However, with less monitoring points, the false alarm rate may increase as a consequence. In this paper, we study the fundamental tradeoff between the number of monitoring nodes and the false alarm rate in the wireless sensor networks. Specifically, we propose fully distributed monitoring algorithms, to build up a poller-pollee based architecture with the objective to minimize the number of overall pollers while bounding the false alarm rate. Based on the established monitoring architecture, we further explore the hop-by-hop aggregation opportunity along the multihop path from the polee to the poller, with the objective to minimize the monitoring overhead. We show that the optimal aggregation path problem is NP-hard and propose an opportunistic greedy algorithm, which achieves an approximation ratio of $\frac{5}{4}$. As far as we know, this is the first proved constant approximation ratio applied to the aggregation path selection schemes over the wireless sensor networks.**

I. INTRODUCTION

As sensor nodes usually operate in an unattended, harsh environment, they are prone to failure and may run out of battery [1], [17]. To make sensor network reliable as well as adaptable, sensor status (such as liveness, density estimation, residue energy, etc.) has to be closely monitored and made known to the sink, which can promptly react to sensor status changes.

In distributed systems, the only way to learn the status of a node is through receiving messages from the node. For example, in IP networks, the poller-pollee structure [10] has been widely used for network management, where some specialized nodes are called *pollers* and the other nodes are called *pollees*. Each poller monitors its pollees by actively sending a "ping" message and then waiting for the reply or by passively waiting for the pollees to send messages periodically. Compared with the wired networks, designing monitoring mechanisms for sensor networks has more challenges. One major challenge is that instead of over a fixed topology, sensors need to self-organize themselves into a monitoring architecture in a distributed manner. Therefore, distributed monitoring is a critical and challenging issue.

In this paper, we propose solutions to address the challenge specific to sensor networks, to design a fault tolerant, energy efficient monitoring system in a distributed manner. The whole architecture is build upon the poller-pollee structure, where

sensors self-organize themselves into two tiers, with pollees in the lower tier and pollers in the upper tier. The pollees send status reports to the pollers along multihop paths, during which the intermediate nodes do the aggregation to reduce the message overhead. Each poller makes local decisions based on the received aggregated packets, and forwards its decision towards the sink.

When building the monitoring architecture, we focus on the fundamental tradeoff between the number of monitoring nodes (i.e., pollers) and the false alarm rate. Most of the previous work targets at minimizing the number of pollers only, because selecting more pollers will enhance the difficulty of tracking the status of each poller and thus increase the network management cost [10]. However, in a lossy environment, the false alarm rate can be adversely affected by a smaller number of pollers. For example, if the number of pollers is too small, some pollees will be too far away from the poller, and then the chance of link transient failure will be higher and the false alarm rate will be larger. To balance the tradeoff between the number of pollers and false alarm rate, we propose a distributed deterministic algorithm, which uses two parameters k_1, k_2 to guide a better distribution of poller and pollee; i.e., no two pollers are less than k_1 hops away from each other, and no pollee is more than k_2 hops away from its poller. This property enables us to minimize the number of pollers while bounding the maximum false alarm rate. We discuss how to set up these parameters and further reduce the message overhead based on a randomized technique.

To increase the energy efficiency and reduce the monitoring overhead, we take the hop-by-hop aggregation opportunities in sensor networks. When pollees, which can be multiple hops away from the poller, send status reports to their pollers, the status reports can be aggregated to reduce the number of packets needed. It is nontrivial to determine which aggregation path should be used in order to achieve better aggregation. For example, as shown in Fig. 1, the *aggregation ratio* (s), which is defined as the number of status reports that can be aggregated into one packet, is 2. Each of the four pollees sends a report to the same poller periodically. Fig. 1(a) shows that without aggregation, 4 packets will be needed for each period, with a cost of 6 packet-hops. If the reports are aggregated along the path shown in Fig. 1(b), all 4 reports can be packed into 3 packets, with a cost of 5 packet-hops. If the aggregation path of Fig. 1(c) is used, only 2 packets are needed, with a total of 4 packet-hops. In this paper, we formulate the selection of the optimal aggregation path to minimize the transmission energy as a NP-hard problem, and prove that an opportunistic greedy

forwarding scheme has an approximation ratio of $\frac{5}{4}$. We also prove that if the pollee is within 2 hops of the poller, i.e., $k_2 = 2$, the bounded ratio is $1 + \frac{s-1}{s^2}$.
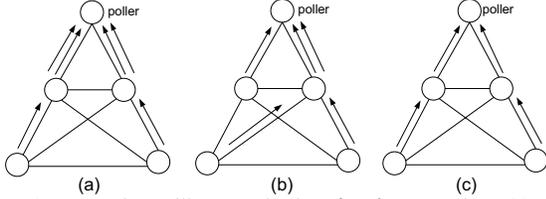

Fig. 1. An example to illustrate the benefit of aggregation: (a) without aggregation (b) aggregation path I (c) aggregation path II, where each line denotes a physical link, and each arrow denotes a packet transmission over one hop (packet-hop)

The rest of the paper is organized as follows. Section II presents the poller-pollee based monitoring architecture. Section III formulates the minimum poller selection problem and proposes the distributed algorithms. Section IV focuses on the optimal aggregation path problem, and studies the greedy algorithm with constant approximation ratio. Performance evaluations are done in Section V. Section VI overviews the related work and Section VII concludes the paper.

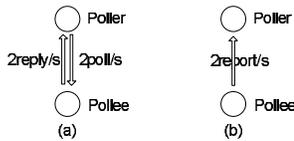## II. THE POLLER-POLLEE BASED MONITORING ARCHITECTURE

Fig. 2. The poller-pollee structure (a) reactive mode (b) proactive mode

Fig. 2 shows the two widely used operational modes of the poller-pollee structure, where each poller can either poll the pollee and wait for a reply (i.e., reactive mode) or let the pollee send reports periodically (i.e., proactive mode). As the proactive mode cuts the monitoring traffic by half compared with the reactive mode, we use the proactive mode in this paper.

Each Link in Fig. 2 is a logical link, which could be multiple hops of physical links. As a result, the status reports destined to the same poller have the opportunity to be aggregated at every intermediate node. To do aggregation, each intermediate node may have to wait for reports to arrive from the downstream nodes. Due to the regular pattern of the monitoring traffic, the aggregation rules can be well defined without adding extra delay. The poller, pollees, and the physical link between them form a tree. If a node is at the edge of the tree, it is called an *edge node*; otherwise it is called a *non-edge node*.

Let t denote the polling time interval. For every time t, each pollee sends a report to the poller. Suppose the poller needs time T_d, referred to as the detection time, to detect a pollee failure. A false alarm occurs if a pollee does not fail but the poller misses all its reports within T_d [7], [12].

The *Aggregation Aware Monitoring* can be simply stated as follows. Each pollee schedules a report to the same poller every t; each non-edge pollee collects reports from each of its children and sends an aggregated report (including its own)

to the same poller every t, with the aggregation ratio s; each poller makes the decision about each pollee's status every T_d, and informs the sink when necessary. It is required $t \leq T_d$.
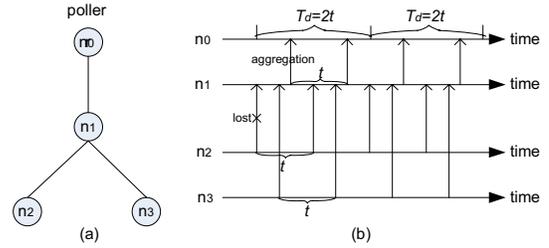

Fig. 3. An example of the aggregation assisted monitoring scheme. (a) topology (b) unsynchronized schedule. The vertical arrow line denotes the event of packet transmission/reception.

Fig. 3 uses an example to illustrate the poller-pollee based monitoring. Fig. 3(a) shows the topology, where n_0 is the poller and n_1, n_2, n_3 are the pollees. Fig. 3(b) shows the schedule at each node, where the vertical arrow line denotes the event of packet transmission/reception. The time axes is divided into slices of polling interval t, and different nodes do not have to be synchronized. Every t, n_2, n_3 send reports to n_1, which aggregates the received reports with its own into one packet (assume $s \geq 3$). With detection timer $T_d = 2t$, the poller will receive two reports every T_d. Suppose one report from n_2 is lost, the poller may still receive another report during the next polling interval and will not have a false alarm about n_2. Thus, there is a tradeoff between false alarm rate and detection delay. To reduce the false alarm rate, the detection timer T_d should be increased, and vice versa.

The false alarm results from the lossy nature of the wireless links. The failure characteristics of the wireless link has been studied in [9] by analyzing the packet traces over the real sensor testbed. Due to the observed bursty and transient error pattern, the wireless link can be modeled as a continuous time Markov chain [15], [8], and we have established the relationship between the false alarm rate and distance [12]. As our work focuses on how to build a poller-pollee architecture, it is independent of the underneath link model. Therefore, we take the result of [12] and apply it here. Assume the link failure rate is f_l, and use $F(h, T_d)$ to denote the false alarm rate when the pollee is h hops away from the poller, and the detection timer at poller is T_d. Fig. 4 shows the false alarm rate as a function of the number of hops from poller to pollee when $f_l = 0.05$. As can be seen, the false alarm rate increases as h increases, since longer path is more vulnerable to failure. As T_d increases, the false alarm rate decreases. This is also consistent with the result of [7].

III. THE MINIMUM POLLER SELECTION PROBLEM

In this section, we formulate the poller selection problem as NP-hard and propose distributed algorithms.

A. Problem Formulation

We consider a network of n sensors, where all sensors are capable of being either pollers or pollees. At first hand, we
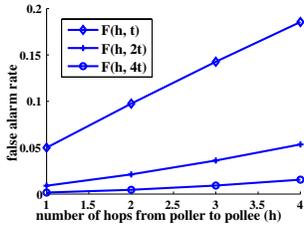
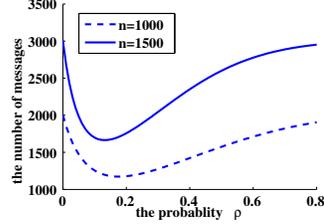Fig. 4. Relationship between the false alarm rate and the distance

Fig. 5. Relationship between ρ and message overhead

1) The Randomized Algorithm: The randomized algorithm is presented as a baseline for comparison. Each node elects itself as a poller with probability ρ. Pollers then announce their poller status within k hops. Sensor nodes that did not elect themselves as pollers will be pollees. The randomized algorithm is very simple, yet it may produce some pathological scenario where multiple pollers may cluster together in some area and no poller exists in some other area. To address this problem, we propose a deterministic algorithm.

Algorithm 1 Deterministic Poller Selection Algorithm

Input: a graph $G(N, E), k_1, k_2$
Output: a Poller Set S_{er}, a Pollee Set S_{ee}
Procedure: $Determine(k_1, k_2)$
1: Initialize the status and the timer
2: Broadcast locally to get k_1-hop neighborhood information
3: **if** timer not expired **then**
4:     **if** id is the smallest among k_1 hop unlabeled neighbors **then**
5:         broadcast $pollerID = id$ within k_2 hops, exit
6:         /*$S_{er} = S_{er} \cup \{id\}$*/
7:     **end if**
8:     wait until a packet is received or the timer is expired
9:     **if** $pollerID$ received **then**
10:        broadcast $polleeID$ within k_1 hops, exit
11:        /*$S_{ee} = S_{ee} \cup \{id\}$*/
12:     **end if**
13:     **if** $polleeID$ received **then**
14:        update the unlabeled List within k_1 hops, reset timer and go to 4
15:     **end if**
16: **else**
17:     go to 5
18: **end if**

---

want to select the minimum number of nodes as pollers so that the management cost of pollers can be minimized. On the other hand, if the number of pollers is too small, some pollees will be many hops away from a poller, thus increasing the false alarm rate. Therefore, our goal is to strike a balance between the number of pollers and false alarm rate. Since the pollers may also fail, we associate each pollee with $\omega \geq 1$ pollers. Each pollee maintains pointers to the different pollers but sends status report to only one poller at a time. When the poller fails, the associated pointers should be outdated and the next poller on the list will be used.

We formulate the poller selection problem *minPL*, with the objective to minimize the number of pollers while limiting the maximum false alarm rate as follows.

Given a network graph, the error rate of each link, determine (1) which nodes are pollers and which nodes are pollees, (2) a many-to-many mapping where each node is associated with ω pollers, to minimize the total number of pollers while limiting the maximum false alarm rate of pollees.

Theorem 1: The problem *minPL* is NP-Hard.

Proof: The problem minPL can be proved to be NP-hard via a reduction from the minimum k-hop dominating set problem [4], which can be seen as a special case of minPL when $\omega = 1$ and the link failure rate is constant. This is because according to Fig. 4, different hop numbers between pollee and poller correspond to different false alarm rates. Therefore, satisfying the constraint of maximum false alarm rate is equivalent to limiting the maximum number of hops from pollee to poller. ∎

B. Distributed Poller Selection Algorithms

The construction of poller-pollee structure shares some similarity with the traditional clustering scheme, where a poller is similar to a cluster head. However, there is fundamental difference between them. First, the traditional clustering schemes are single-hopped, but the pollee should be within some bounded hops of its poller. Second, with multihops between the poller and pollees, aggregation is used to reduce the monitoring traffic, which is not considered in clustering schemes. Third, each pollee may be associated with $\omega \geq 1$ pollers to be fault tolerant, whereas each cluster member only has one cluster head. Thus, the traditional clustering scheme is only a special case of the single-hop poller-pollee structure with $\omega = 1$. Below, we first propose a distributed deterministic poller selection algorithm, and then present a hybrid algorithm to further reduce the message overhead.

2) The Deterministic Algorithm: The proposed deterministic algorithm is based on the distributed *maximal independent set (MIS)* algorithm [2]. An *Independent Set* is a subset of nodes among which there is no edge between any two nodes. The set is a MIS if no more edges can be added to generate a bigger independent set. In the deterministic algorithm, the concept of MIS is extended to the multihop environment. Two parameters k_1, k_2 are used to govern the distribution of the pollers and pollees, to ensure that no two pollers are less than k_1 hops from each other and no pollee is more than k_2 hops from its poller. That is, the poller set S_{er} is a k_1-hop MIS, in which no two nodes are less than k_1 hops away from each other.

Since parameters k_1, k_2 control the geometrical properties of the poller and pollee distribution, they should be determined beforehand according to user's demand. First, given the constraint of false alarm rate, k_2 can be determined, based on the relationship between the false alarm and the hop distance such as in Fig. 4. Thus, bounding the maximum false alarm rate becomes equivalent to bounding the maximum distance from pollee to the poller. Second, after k_2 is determined, k_1 can be selected. Although a large k_1 could reduce the number of pollers selected but some pollee may not find ω pollers within k_2 hops. Therefore, an appropriate k_1 should be chosen to strike a balance between the number of pollers and the number of unlabeled nodes. This can be achieved by the experiments, as in Section V.

Given k_1, k_2, Algorithm I lists the pseudo code of the deterministic poller selection algorithm. At the outset, each node needs to obtain the k_1-hop neighborhood information by
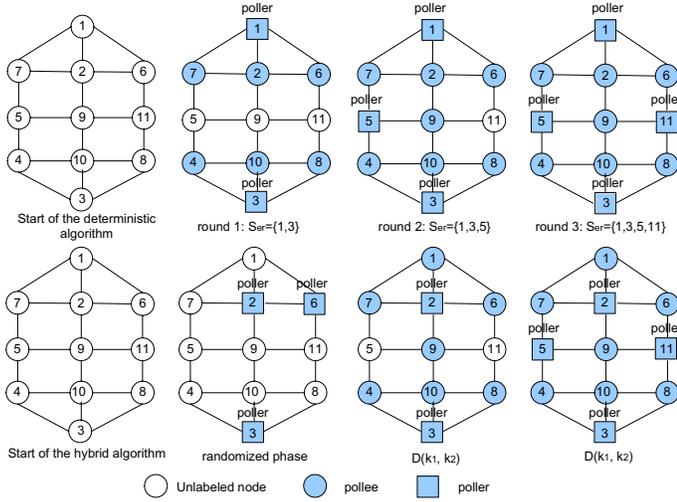
Fig. 6. A numerical example. The deterministic algorithm (above) runs in three rounds, exchanging 22 messages. The hybrid algorithm (below) has a randomized phase and two deterministic phases, exchanging 10 messages.

a localized broadcast. Then the algorithm proceeds in rounds. In each round, if a node has the smallest id among the k_1-hop unlabeled neighbors, it elects itself as belonging to the poller set S_{er}, and broadcasts $pollerID = id$ within k_2 hops. Nodes that are not yet labeled but received the poller declaration will label themselves as pollees, and broadcast $polleeID = id$ within k_1 hops. After that, a new round will start, during which the algorithm is executed among the remaining unlabeled nodes. This process repeats until all nodes are labeled either as poller or pollee.

The upper diagram of Fig. 6 uses a small network of 11 nodes to explain the deterministic algorithm, assuming $k_1 = k_2 = 1$. In the first round, node 1 and node 3 elect themselves as poller since their id is the smallest among the 1-hop neighborhood, and their 1-hop neighboring nodes 2,4,6,7,8,10 are recruited as pollees. Then the labeling process will repeat among the remaining nodes until everyone is labeled. Thus, in the second round, node 5 is labeled as poller and node 9 is labeled as pollee. In the third round, node 11 is labeled as poller.

The message complexity of Algorithm I can be computed as follows. Assume there are n nodes. At first, each node needs to broadcast within k_1 hops to get the neighborhood information (line 2). After that, each poller needs to broadcast within k_2 hops (line 5) and each pollee needs to broadcast within k_1 hops (line 10). Suppose $k = k_1 = k_2$, each node broadcasts within k hops exactly twice during the execution of the algorithm. Further suppose d is the node density, and r is the node communication range. The message complexity can be estimated by $O(2*n*d*(kr)^2) = O(k^2 nd)$.

It is possible to set k_1 as a fractional number. For example, when $k_1 = 1.2$, it means that each node chooses $\lfloor k_1 \rfloor = 1$ with probability 0.8 and chooses $\lceil k_1 \rceil = 2$ with probability 0.2. As k_1 controls the distance between the neighboring pollers, allowing k_1 to be a fractional number will provide more flexibility to control the number of pollers, which can be seen from Section V. Then, Algorithm 1 needs to be extended as follows. First, set $k_1' = \lfloor k_1 \rfloor$ with probability $\lceil k_1 \rceil - k_1$ and

set $k_1' = \lceil k_1 \rceil$ with probability $k_1 - \lfloor k_1 \rfloor$. Second, each node elects itself as poller among the k_1'-hop unlabeled neighbors (line 4). Third, each node broadcasts id within $\lceil k_1 \rceil$ hops (line 5,10), but updates the unlabeled list within k_1' hops (line 14).

Algorithm 2 Hybrid Poller Selection Algorithm

Input: a graph $G(N, E)$, k_1, k_2, ρ
Output: a Poller Set S_{er}, a Pollee Set S_{ee}
Procedure: $Hybrid(k_1, k_2)$
1: Initialize the node label, timer, $S_{er} = \phi, S_{ee} = \phi$
2: generate a random number $\sigma \in (0, 1)$
3: **if** $\sigma < \rho$ **then**
4:　　$S_{er} = S_{er} \cup \{id\}$
5: **end if**
6: **if** $id \in S_{er}$ /*node id is temporarily labeled as poller*/ **then**
7:　　execute the deterministic algorithm $Determine(k_1, k_2)$
8:　　/*S_{er} and S_{ee} are updated*/
9: **end if**
10: **if** node id is unlabeled **then**
11:　　wait until a packet is received or the timer is expired
12:　　**if** $pollerID$ received **then**
13:　　　　label itself as pollee /*$S_{ee} = S_{ee} \cup \{id\}$*/
14:　　**end if**
15: **end if**
16: **if** $id \in \{N - S_{er} - S_{ee}\}$ /*node id is not labeled*/ **then**
17:　　execute the deterministic algorithm $Determine(k_1, k_2)$
18: **end if**

---

3) The Hybrid Algorithm: In the deterministic algorithm, each node needs to broadcast within k hops twice, assuming $k = k_1 = k_2$. As k becomes larger, the message complexity increases dramatically. To reduce this message overhead, we propose a hybrid algorithm which combines the randomized algorithm and the deterministic algorithm. As shown in Algorithm 2, the algorithm starts with a randomized phase, during which each node labels itself as a temporary poller with a probability ρ. After that, the deterministic algorithm is executed among the temporary pollers. If two pollers are less than k_1 hops away from each other, one of them will change its role from poller to pollee and the other one will confirm itself as poller. After receiving the confirmed $pollerID$, the unlabeled nodes within k_2-hop range of the confirmed poller will be recruited as pollees. To implement this, a field in the packet header needs to be reserved to differentiate the confirmed $pollerID$ from the temporary $pollerID$. After receiving the confirmed $pollerID$, the unlabeled nodes change the status to pollee. Finally, the deterministic algorithm is executed among the set of unlabeled nodes, i.e., $\{N - S_{er} - S_{ee}\}$, to have all the nodes labeled.

Compared with the deterministic algorithm, the hybrid algorithm has much less message overhead. First, due to the use of the randomized phase, each node does not have to make the initial $k_1 - hop$ broadcast. Second, when the deterministic algorithm is executed for the first time (line 7), only the temporary pollers are involved. Third, after receiving the confirmed $pollerID$, many unlabeled nodes become pollees who are refrained from broadcast. Thus, only a small portion of the nodes still remain unlabeled and execute the deterministic algorithm for the second time (line 17). In the following, we give a detailed analysis of the message overhead, based on which the optimal value of ρ can be derived to minimize the message overhead.

For ease of illustration, we first assume $k_1 = k_2 = k$,

and then extend it to the general case of $k_1 \neq k_2$. First, the randomized phase introduces zero message overhead. As a result, on average a fraction ρ of the nodes are labeled as pollers and execute the deterministic algorithm for the first round (line 7). Among the $n\rho$ temporary pollers, a fraction of nodes, say, α, are confirmed as pollers, and a fraction $1 - \alpha$ of the nodes become pollees. During the execution of the first-round deterministic algorithm, the $n\rho\alpha$ confirmed pollers will recruit unlabeled nodes within k hops as pollee. After that, only a fraction, say, β, of the $n(1 - \rho)$ unlabeled nodes still remain as unlabeled and execute the second round (line 17). Therefore, the total number of nodes involved in the two rounds of the deterministic algorithm is $n\rho + n(1 - \rho)\beta$, with all the other nodes recruited as pollee by the $n\rho\alpha$ confirmed pollers. Further use $M(k)$ to denote the message complexity of k-hop broadcast. Since each participating node needs to broadcast twice within k hops, the overall message complexity is:

$$2[n\rho + n(1 - \rho)\beta]M(k) \quad (1)$$

β is the fraction of $n(1-\rho)$ unlabeled nodes that do not fall in the k-hop range of any of the $n\rho\alpha$ confirmed pollers. For a node n_1, the probability that it falls within the k-hop range of node n_2 is $\frac{\pi k^2 r^2}{a^2}$, where a^2 denotes the area size. Thus,

$$\beta = (1 - \frac{\pi k^2 r^2}{a^2})^{n\rho\alpha} \quad (2)$$

α denotes the fraction of nodes that are to be labeled as poller if $n\rho$ uniformly distributed nodes run the deterministic algorithm. When two nodes are within k hops, one of them will refrain from being a poller with half chance. Since there are total $n\rho$ nodes, the probability for each node to become a poller is

$$\alpha = (1 - \frac{\pi k^2 r^2}{2a^2})^{n\rho - 1} \quad (3)$$

Substituting Eqn. 3 and Eqn. 2 into Eqn. 1 will give us the message complexity when $k_1 = k_2$.

Fig. 5 numerically evaluates the effect of varying the number of nodes n and the probability ρ on the amount of message overhead, when $a = 20, r = 1, k_1 = k_2 = 1$. It is shown that each curve has an optimal ρ corresponding to the minimum message overhead. In addition, the optimal value of ρ reduces as the number of nodes n increases. For example, ρ is about 0.2, 0.15, when $n = 1000, 1500$, respectively.

Fig. 6 uses a simple network to compare the deterministic algorithm and hybrid algorithm. In the hybrid algorithm (lower diagram), the randomized phase selects three nodes, i.e., nodes 2,3,6, as temporary pollers, among which the deterministic algorithm is executed. Node 6 resigns from the role of poller and becomes pollee because its id is larger than its neighbor node 2, which is also a poller. As a result, nodes 2,3 are confirmed as pollers and broadcast locally to recruit nodes 1,7,9,4,8,10 as pollees. After that, only nodes 5,11 still remain unlabeled, among which the second round of the deterministic algorithm is executed. In comparison of the message overhead, the deterministic algorithm has $2 \times 11 = 22$ message exchanges, but the hybrid algorithm only has $2 \times (3+2) = 10$ message exchanges. The message complexity is reduced by over 50% in this example. As k_1, k_2 become larger and the network grows to a larger scale, the message saving will be more significant.

After pollers are determined, each pollee needs to select up to ω pollers based on the received announcements. The selection could be simply based on criteria such as the distance to the poller. Each pollee maintains a list of pointers to the ω pollers, and each pointer links to the routing entry (e.g., distance, next hop) of a distinct poller. After a poller fails, each pollee will update its active poller list, remove the stale entry and choose the next poller on the list. As more pollers fail, new pollers can be elected by running Algorithm 1 among the neighboring pollees of the failed poller.

IV. THE OPTIMAL AGGREGATION PATH PROBLEM

A. The Problem Formulation

After the poller-pollee relationship is established, each pollee starts to periodically send a report to its poller. Along the multihop path from pollee to poller, each non-edge pollee collects reports from its downstream children and sends an aggregated report (including its own) to the poller. As demonstrated in Fig. 1, a pollee may have multiple paths to the poller, which results in different energy efficiency. Assume transmitting one packet over one hop consumes one unit of energy. The problem of finding the optimal aggregation path (*optPH*) can be formulated as follows.

In a network graph, given the poller-pollee relationship, a constant aggregation ratio, each node needs to send a report to its poller periodically. For each poller, find the optimal aggregation paths from all its pollees, such that the total energy consumption is minimized.

When the aggregation ratio s is infinite, it is called perfect aggregation. That is, infinite number of reports can be aggregated into one packet. It is not hard to see that the optimal solution of this special case is the minimum spanning tree. For the general case where the aggregation ratio is finite, it can be proved that the problem of optPH is NP-hard, which is consistent with the findings in [14], [3]. The proof is omitted here due to the limited space.

B. A Greedy Algorithm And Its Approximation Ratio

Although the problem of selecting the optimal aggregation path is NP-hard, we can design some heuristics. A simple greedy algorithm is to arbitrarily select the next hop to forward, as long as it is on the shortest path to the poller, and opportunistically aggregate the status reports at the intermediate nodes. This seemingly simple algorithm has surprisingly good performance, with a tight bound of $\frac{5}{4}$. That is, the energy consumption resulted from the greedy algorithm will not be more than $\frac{5}{4}$ times the optimum solution. Before proving the approximation bound of the greedy algorithm for the optPH problem, we first define some terminologies.

Within a polling domain, the shortest paths from all the pollees to the poller of the domain form a tree, which may be partitioned into different levels. A node is said to be at *level* i ($i \geq 0$) if it is i hops away from the sink (i.e., the poller). Therefore, the sink is the only node at level 0. A rooted tree is called a *level-connected tree* if the nodes
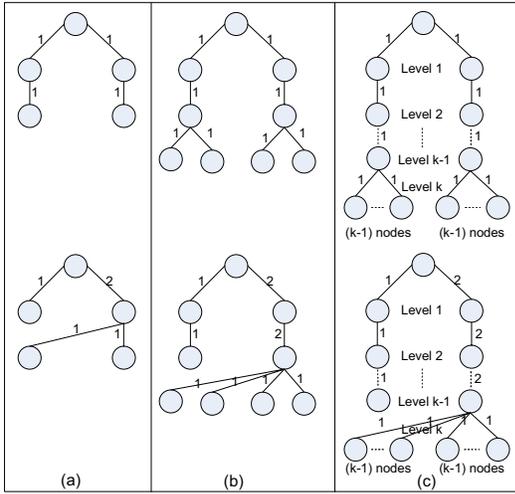
Fig. 7. Worst-case topology with the best strategy (above) and worst strategy (below). (a) 2-level connected tree, $s = 2$. (b) 3-level connected tree, $s = 4$. (c) k-level connected tree, $s = 2(k - 1)$.

belonging to the neighboring levels are connected with each other. Specifically, a k-level-connected tree is composed of the nodes at level-i, $i = 0, 1...k$. Given a tree topology, the set of aggregation paths that consume the least/most energy is called the *best/worst paths*, denoted as H_b and H_w, respectively. Correspondingly, the aggregation strategy that can produce the best/worst paths is called the *best/worst strategy*. Given a topology, the ratio of energy consumption between the worst and best paths determines the *performance bound/ratio* of the topology, denoted as δ. Denote the energy consumption of the best path H_b and worst path H_w as E_b, E_w, then we have $\delta = \frac{E_w}{E_b}$. Given an algorithm, the maximum performance bound/ratio among all possible topologies, denoted as δ_{max}, is called the *approximation ratio* of the algorithm. The *worst-case topology* is the topology whose performance bound equates the approximation ratio of the algorithm.

To have an intuitive idea about how to calculate the performance bound, we first show some topologies in Fig. 7, whose performance bound is $\frac{5}{4}$. We can later prove that $\delta = \frac{5}{4}$ is the maximum performance bound among all topologies, so these classes of k-level-connected tree are actually the worst-case topology. In Fig. 7, the worst-case k-level connected tree is constructed by having each level-i, $i = 1...k - 1$ with 2 nodes and level-k with $2(k - 1)$ nodes, where Figs. 7(a)(b) are the special cases when $k = 2, 3$ and Fig. 7(c) is the general case. Note that in each level-connected tree, nodes at the neighboring levels are all connected with each other. But for the clarity purpose not all the links are shown in Fig. 7, with only the aggregation paths depicted. The number shown by each link indicates the number of packet transmissions over that link. In Fig. 7, the upper diagram represents the best aggregation paths, and the lower diagram represents the worst aggregation paths. A further calculation shows that the energy consumptions of the best and worst paths are $4(k - 1)$ and $5(k - 1)$ in Fig. 7(c). With $s = 2(k - 1)$, we have $\delta = \frac{5}{4}$.

In the rest of this section, we will prove that $\delta = \frac{5}{4}$ is the maximum performance bound among all topologies, and thus the approximation ratio of the greedy algorithm. As all the shortest paths constitute a tree and any tree topology is

a subset of some level-connected tree, we can restrict our attention to the level-connected tree. The following theorem gives the approximation ratio for the 2-level-connected tree, which will be used to derive the approximation ratio for the arbitrary level-connected tree.

Theorem 2: For a 2-level-connected tree, the approximation ratio of the opportunistic greedy algorithm is no greater than $1 + \frac{s-1}{s^2}$, where s is the aggregation ratio.

Proof: Given a 2-level-connected tree, the energy spent by sending the reports from the level-2 nodes to the level-1 nodes is fixed, i.e., equal to the number of level-2 nodes. The problem optPH becomes how to minimize the energy spent between the level-1 nodes and the root by packing the level-2 reports into a minimum number of level-1 packets. We give the *best* and *worst* aggregation strategy as follows.

Best strategy: packing/sending every $s - 1$ level-2 reports into a single level-1 node until all level-1 nodes have a packet of s (including its own) reports constructed without residue room left. All the remaining level-2 reports are then packed into a single level-1 node.

Worst strategy: packing/sending all level-2 reports into a single level-1 node.

Excluding its own report, each level-1 node has $s - 1$ vacancies in the first packet. The best strategy tries to fully utilize all the vacancies, while the worst strategy tries to use as many packets as possible by packing all status reports aggressively into a single node. There may be other alternative best or worst strategies, but it is sufficient to find one of them for our proof. Taking Fig. 8(b) as an example, where $s = 3$, the best strategy packs every 2 level-2 reports into a level-1 node, and the worst strategy packs all 6 level-2 reports into a single level-1 node. A simple calculation shows that $\delta = \frac{11}{9}$.

Now assume level 1 has l_1 nodes and level 2 has l_2 nodes. Then the worst-case topology must satisfy: $(s - 1)(l_1 - 1) < l_2 \leq (s - 1)l_1$. That is, all but one of the level-1 nodes need to be fully packed. Otherwise, if $l_2 \leq (s - 1)(l_1 - 1)$, some level-1 nodes become edge nodes without children in both the best and worst strategy, and could be removed to increase the performance ratio. In other words, the original topology is not the worst case topology when $l_2 \leq (s - 1)(l_1 - 1)$. On the other hand, if $l_2 > (s - 1)l_1$, $l_2 - (s - 1)l_1$ level-2 edge nodes can be removed to increase the performance ratio. To see this, removing level-2 nodes deceases the number of packets packed at level 1 by *exactly* $\lceil \frac{l_2 - (s-1)l_1}{s} \rceil$ in the best strategy, but decreases by *at most* $\lceil \frac{l_2 - (s-1)l_1}{s} \rceil$ in the worst strategy. Thus, the original topology is not the worst case topology when $l_2 > (s - 1)l_1$.

Therefore, we set $l_2 = (s - 1)l_1 - i$, $i = 0...s - 2$. It is followed that $E_b = l_1 + l_2$, $E_w = \lceil \frac{l_2+1}{s} \rceil + (l_1 - 1) + l_2$. Then,

$$\delta - 1 = \frac{E_w - E_b}{E_b} = \frac{\lceil \frac{l_2+1}{s} \rceil - 1}{l_1 + l_2} \leq \frac{\frac{l_2+1+(s-1)}{s} - 1}{l_1 + l_2}$$

$$= \frac{1}{s} \frac{(s-1)l_1 - i}{sl_1 - i} \leq \frac{1}{s} \frac{(s-1)l_1}{sl_1} = \frac{s-1}{s^2} \quad (4)$$

The performance ratio $1 + \frac{s-1}{s^2}$ is tight, which can be seen from Fig. 8. Specifically, Figs. 8(a)&(b) show the worst-case
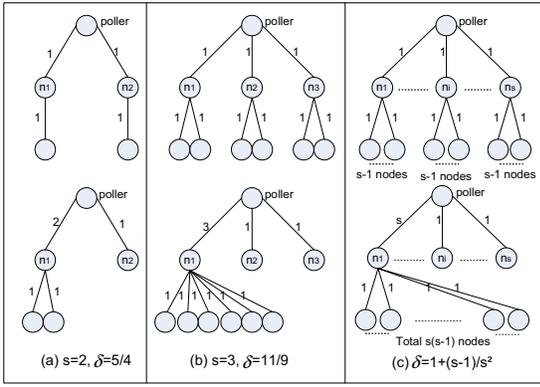
Fig. 8. The derived bound is tight for 2-level-trees: the best strategy (above) and worst strategy (below). The number by the link indicates the number of packets transmitted over this link.
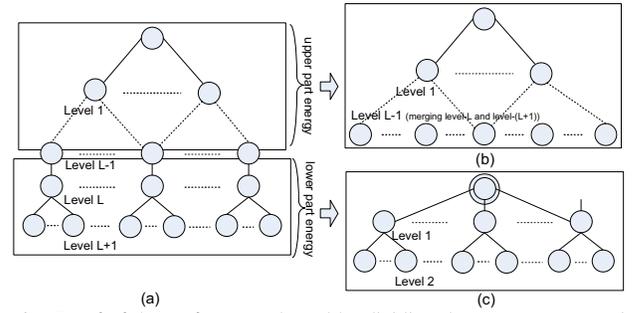

Fig. 9. Proof of the performance bound by dividing the energy consumption into two parts. (a) the original connected tree (b) the upper part consists of the links above level-(L-1), with the nodes at level-L and level-(L+1) expanded onto level-(L-1). (c) the lower part consists of the links below level-(L-1), with the nodes at level-(L-1) and above shrunk into a single virtual node.

topology when $s = 2$ and $s = 3$. For arbitrary s, the worst-case topology can be constructed as in Fig. 8(c), where level 2 has s nodes and level 3 has $s(s-1)$ nodes, then we have

$$\delta = 1 + \frac{\lceil \frac{l_2+1}{s} \rceil - 1}{l_1 + l_2} = 1 + \frac{\lceil \frac{s(s-1)+1}{s} - 1 \rceil}{s(s-1) + s} = 1 + \frac{s-1}{s^2} \quad (5)$$

Theorem 3: The approximation ratio for the arbitrary level■ connected tree is no greater than $\frac{5}{4}$, i.e., $\delta_{max} = \frac{5}{4}$.

Proof: The theorem can be proved by the induction of L, the depth of the tree. First, Theorem 2 shows that the base case for 2-level-connected tree is true, i.e., $\delta_{max} = max(1 + \frac{s-1}{s^2}) = \frac{5}{4}$. Assuming the claim is true for k-level-connected tree, $\forall k \leq L$, we want to prove that $\delta_{max} = \frac{5}{4}$ for (L+1)-level-connected tree.

As shown in Fig. 9(a), we divide the total energy consumption of the (L+1)-level-connected tree into two parts, namely, the upper part consisting of the links from level-(L-1) to level-0 (Fig. 9(b)) and the lower part consisting of the links from level-(L+1) to level-(L-1) (Fig. 9(c)). Because the nodes below level-(L-1) also contribute to the energy consumption of the upper part, we need to expand the nodes below level-(L-1) onto level-(L-1). Then the level-(L-1) nodes in Fig. 9(b) are the coalescence of level-(L-1), level-L and level-(L+1) nodes in Fig. 9(a). On the other hand, The nodes at level-(L-1) and above have no contribution to the energy consumption of the lower part, so they can be safely shrunk into one virtual node. As shown in Fig. 9(c), the virtual node is the root of the 2-level tree. Use E_w^{up}, E_b^{up} to denote the energy consumption of the worst path and the best path in the upper part (Fig. 9(b)), and use E_w^{low}, E_b^{low} to denote the energy consumption of the worst path and the best path in the lower part (Fig. 9(c)). It can be seen that the above expanding and shrinking operation shall not reduce the gap between the energy consumption in the best case and worst case.

Fig. 9(b) is a (L-1)-level-connected tree. By the inductive hypothesis, $\delta_{max} = \frac{5}{4}$ for Fig. 9(b). Therefore

$$E_w^{up} - E_b^{up} \leq (\frac{5}{4} - 1)E_b^{up} = \frac{1}{4}E_b^{up} \quad (6)$$

Fig. 9(c) is a 2-level-connected tree. According to Theorem 2, $\delta_{max} = \frac{5}{4}$ for Fig. 9(c). Then we have

$$E_w^{low} - E_b^{low} \leq (\frac{5}{4} - 1)E_b^{low} = \frac{1}{4}E_b^{low} \quad (7)$$

Combining Eqn. 6 and Eqn. 7, we finally have

$$\delta = \frac{E_w^{up} + E_w^{low}}{E_b^{up} + E_b^{low}} \leq \frac{5}{4} \quad (8)$$

The bound of $\frac{5}{4}$ is tight as seen from Fig. 7, where the worst-case topologies are shown.

V. PERFORMANCE EVALUATIONS ■

In this section, we use simulation to test the parameters and evaluate the proposed algorithms. In the simulation, n sensors are randomly deployed in a 20×20 square area. Each pollee is associated with up to ω pollers, which are used for fault tolerance. Each sensor sends a status report to its poller every t, with the reports aggregated at the intermediate nodes. The sensor transmission range is 1, and transmitting one packet over one hop consumes one unit of energy. The link failure is modeled as a continuous-time markov chain with an average failure rate of $f_l = 0.05$ and a detection period $T_d = 2t$. The experiments are done over a customized C++ simulator.

A. Parameter Setting

Three parameters need to be determined, k_1, k_2 for the deterministic algorithm and ρ for the hybrid algorithm. In Section II, Fig. 4 gives the relationship between the false alarm rate and the distance from the pollee to the poller when the average link failure rate is known. As a result, given the constraint of the maximum false alarm rate, k_2 can be determined to limit the number of hops from the pollee to the poller. Suppose the false alarm rate is required to be less than 4%, then $k_2 \leq 3$ based on Fig. 4. We will set $k_2 = 3$ in the following experiments unless otherwise specified, based on which k_1 and ρ are chosen.

Fig. 10 shows the effect of k_1. As k_1 increases, there will be less number of pollers because k_1 controls the distance between the neighboring pollers. However, the number of pollees that cannot find ω pollers increases as k_1 increases. For example, as k_1 increases from 1 to 1.8, the number of pollees that cannot find 3 pollers ($\omega = 3$) increases from 0 to 210. This is because with less number of pollers, it is less likely for a pollee to find ω pollers within k_2 hops. As can be seen, if $k_1 = 1$, all pollees can find three pollers. However, if $k_1 = 1.8$, over 20% pollees cannot find three pollers. We thus set $k_1 = 1$ in the following experiments, without otherwise specified. Note that k_1 may be a fractional number.
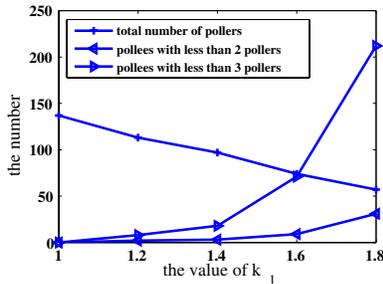
Fig. 10. The effect of k_1 on the number of pollers and the number of pollees that cannot find ω pollers (n=1000)
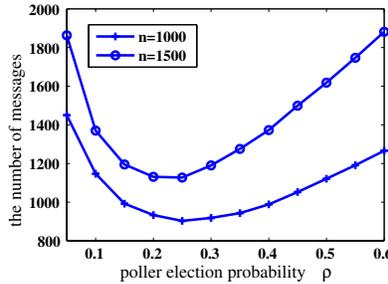
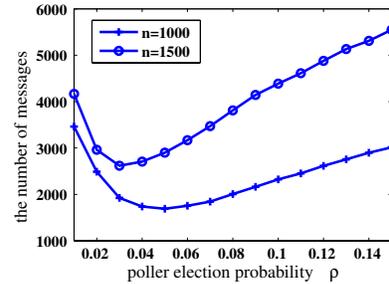Fig. 11. The effect of probability ρ on the message overhead in the hybrid algorithm ($k_1 = 1, k_2 = 1$)

Fig. 12. The effect of probability ρ on the message overhead in the hybrid algorithm ($k_1 = 1, k_2 = 3$)

In the hybrid algorithm, each node first elects itself as poller with probability ρ. Fig. 11 shows how the value of ρ affects the message overhead when $k_1 = k_2 = 1$. Compared with the analytical result in Fig. 5, the same trend is observed in Fig. 11, where the message overhead first drops then rises as ρ increases. The point with the least number of messages corresponds to the optimal ρ^*. For example, when $n = 1000, 1500$, ρ^* is around 0.25, 0.2 in the simulation, and around 0.2, 0.15 in the analysis. The little mismatch of the theoretical result may be explained by the boundary effect of the finite field in reality. That is, in the analysis it is assumed that the sensing range of a node falls within the field of the deployment, but this is not true for the nodes on the boundary, which will affect the accuracy of the analytical result.

Similarly, Fig. 12 shows the relationship between the message overhead and ρ when $k_1 = 1, k_2 = 3$. It can be observed that at the optimal point, ρ^* is around 0.05, 0.03 when $n = 1000, 1500$, which is much smaller than the case when $k_1 = k_2 = 1$. This is because with larger k_2, a smaller number of pollers are able to cover most of the unlabeled nodes. In the following, we will use the optimal ρ^* corresponding to the different n, k_1, k_2.

B. Comparison of Different Schemes

In this section, we compare the geometrical property of the hybrid algorithm and the randomized algorithm. Fig. 13 shows snapshots of the poller-pollee distribution after running the randomized and hybrid algorithm among 100 nodes in a 5×5 field. The whole network is connected but for clarity purpose we only show the links within each polling domain. Fig. 13(a) shows that the randomized algorithm produces a scenario where the poller may be isolated (e.g., node 79 in the up-left corner), clustered together (e.g., node 86,39,45 on top), or the pollee (e.g., node 92 in the bottom-left corner) is too far away from its poller. However, after running the deterministic algorithm in Fig. 13(b), the hybrid algorithm fixes these problems. For instance, in Fig. 13(b) there is on isolated poller and no poller is within one hop of each other and no pollee is more than 2 hops away from its poller.

Figs. 14, 15 further compare the hybrid algorithm with the randomized algorithm in terms of statistical geometrical property and false alarm rate. For fair comparison, we set $\rho = 0.75, 0.065, k = 6$, when $n = 1000, 1500$, in the randomized algorithm, to select the same number of pollers as in the hybrid algorithm. As seen in Fig. 14, the distance

between pollers and pollees is bounded by 3 hops in the hybrid algorithm, but spans up to 6 hops in the randomized algorithm. About 8% of the pollees in Fig. 14(a) are more than 3 hops away from their pollers, so the constraint of the false alarm rate cannot be met in the randomized algorithm.

Fig. 15 shows that the hybrid algorithm outperforms the randomized algorithm in terms of the average false alarm rate, which is calculated by averaging the sum of the false alarm rate over all the nodes. It can be seen that the average false alarm rate of the hybrid algorithm is about 20% or 30% smaller than that of the randomized algorithm, when $n = 1000$ or 1500, respectively.

Both the deterministic algorithm and the hybrid algorithm have provable distribution property – no poller is less than k_1 hops away from each other, and no pollee is more than k_2 hops away from its poller. However, Fig. 16 shows that the hybrid algorithm substantially reduces the number of messages. The reduction is about 60% and 80%, when $k_1 = 1$ and $k_1 = 2$, respectively. The benefit is thanks to the randomized phase adopted by the hybrid algorithm.
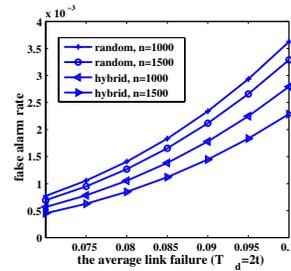

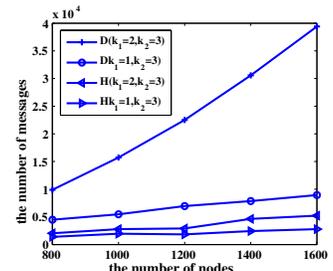Fig. 15. Comparison of random algorithm with hybrid algorithm

Fig. 16. Comparison of deterministic algorithm with hybrid algorithm

VI. RELATED WORK

While a lot of research in sensor network focuses on the field or target monitoring [21], [11], little attention has been given to the monitoring of sensor network itself. In [7], [16], two similar distributed failure detectors were proposed independently for wireless sensor networks, where each node is collaboratively monitored by its one-hop neighbors. These schemes can only detect node failure, but more general status such as residue energy and coverage cannot be monitored. In [22], data aggregation had been used to obtain a global abstraction of the sensors' residue energy, but only when specific continuous energy dissipation models are assumed. More recently, a local monitoring infrastructure is proposed
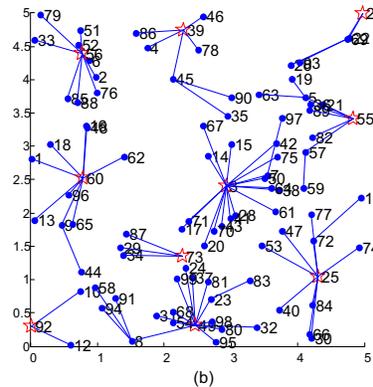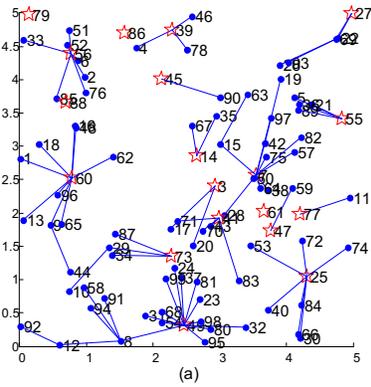
Fig. 13. Snapshots of poller-pollee distribution, where the star denotes the poller and the dot denotes the pollee: (a) randomized algorithm, $\rho = 0.2$, (b) hybrid (randomized + deterministic) algorithm, $\rho = 0.2, k_1 = 1, k_2 = 2$
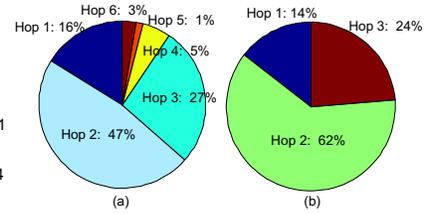

Fig. 14. Distribution of the distance between pollers and pollees: (a)randomized algorithm (b) hybrid algorithm

in [5]. But their goal is to monitor the transmission over the wireless link for security purpose, instead of from the perspective of fault tolerance. By contrast, our poller-pollee based monitoring architecture can respond to queries about a variety of sensor status tailored to the application demand.

The problem of selecting a subset of nodes to form a backbone has been extensively studied in different contexts, e.g., clustering [19], connected dominating set (CDS) [18], [4], relay node placement [13], etc. The objective of these works is to minimize the cardinality of the selected subset of nodes, but ignore the constraint of false alarm rate, which is crucial in wireless sensor networks. In addition, most of these works are limited in the selection of single-hop, single vantage point (e.g., cluster head, dominator). However, our work focuses on multi-hop multi-poller monitoring architecture construction, where some geometrical properties of the poller-pollee distribution can be guaranteed.

Aggregation path selection problem has been proposed in [14], [3], wherein some heuristics are developed but without performance guarantee. There are also some other aggregation schemes that target for the different application scenarios [6], [23], [20], but none of them guarantees constant approximation ratio. To the best of our knowledge, our result is the first proved constant approximation ratio applied to the aggregation path selection schemes for the wireless sensor networks.

VII. CONCLUSIONS

In this paper, we focus on the distributed design of monitoring and aggregation algorithm for wireless sensor network. Based on the poller-pollee structure we first proposed fully distributed algorithms to select the minimum number of pollers while bounding the false alarm rate. Then a greedy aggregation scheme was proposed to reduce the messages overhead due to monitoring. Theoretical analyses and extensive simulations show that the deterministic algorithm can flexibly control the poller-pollee distribution property to bound the false alarm rate, the hybrid algorithm can reduce the message overhead significantly, and the greedy aggregation scheme decreases the monitoring traffic with a constant approximation ratio of $\frac{5}{4}$.

REFERENCES

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E.Cayirci, "Wireless Sensor Networks: A Survey," *Computer Networks*, March 2002.

[2] S. Basagni, "A distributed algorithm for finding a maximal weighted independent set in wireless networks," in 11th *International Conference on Parallel and Distributed Computing and Systems (PDCS)*, 1999.

[3] R. Cristescu, B. Beferull-Lozano, and M. Vetterli, "On network correlated data gathering," in *INFOCOM*, 2005.

[4] F. Dai and J. Wu, "On constructing k-connected k-dominating set in wireless networks," *Journal of Parallel and Distributed Computing (JPDC)*, 2006.

[5] Dezun Dong, Yunhao Liu, and Xiangke Liao, "Self-monitoring for sensor networks," in *MobiHoc*, 2008.

[6] Kai-Wei Fan, Sha Liu, and Prasun Sinha, "On the potential of structure-free data aggregation in sensor networks," in *INFOCOM*, 2006.

[7] Chih fan Hsin and Mingyan Liu, "A distributed monitoring mechanism for wireless sensor networks," in *WISE*, 2002.

[8] A. Konrad, B. Y. Zhao, A. D. Joseph, and R. Ludwig, "A markov-based channel model algorithm for wireless networks," *Wireless Networks*, vol. 9, pp. 189–199, 2003.

[9] H. Lee, A. Cerpa, and P. Levis, "Improving wireless simulation through noise modeling," in *IPSN*, 2007.

[10] Li (Erran) Li, Marina Thottan, Bin Yao, and Sanjoy Paul, "Distributed network monitoring with bounded link utilization in ip networks," in *INFOCOM*, San Francisco, April 2003.

[11] C. Liu and G. Cao, "Minimizing the cost of mine selection via sensor networks," in *INFOCOM*, 2009.

[12] C. Liu and G. Cao, "An multi-poller based energy-efficient monitoring scheme for wireless sensor networks," in *INFOCOM mini-conference*, 2009.

[13] S. Misra, S. Hong, G. Xue, and J. Tang, "Constrained relay node placement in wireless sensor networks to meet connectivity and survivability requirements," in *INFOCOM*, 2008.

[14] S. J. Park and R. Sivakumar, "Energy efficient correlated data aggregation for wireless sensor networks," *International Journal of Distributed Sensor Networks*, 2008.

[15] S. ROSS, *Stochastic Processes*, John Wiley and Sons, 1996.

[16] Stanislav Rost and Hari Balakrishnan, "Memento: A health monitoring system for wireless sensor networks," in *SECON*, 2006.

[17] L. Su, C. Liu, and G. Cao, "Routing in intermittently connected sensor networks," in *ICNP*, 2008.

[18] Peng-Jun Wan, Khaled M. Alzoubi, and Ophir Frieder, "Distributed construction of connected dominating set in wireless ad hoc networks," *Mobile Networks and Applications*, vol. 9, no. 2, 2004.

[19] O. Younis and S. Fahmy, "Distributed clustering in ad-hoc sensor networks: A hybrid, energy-efficient approach," in *INFOCOM*, 2004.

[20] Bo Yu, Jianzhong Li, and Yingshu Li, "Distributed data aggregation scheduling in wireless sensor networks," in *INFOCOM*, 2009.

[21] W. Zhang and G. Cao, "Dctc: Dynamic convoy tree-based collaboration for target tracking in sensor networks," *IEEE Transactions on Wireless Communication*, 2004.

[22] Jerry Zhao, Ramesh Govindan, and Deborah Estrin, "Residual energy scans for monitoring wireless sensor networks," in *WCNC*, 2002.

[23] R. Zheng and R. Barton, "Toward optimal data aggregation in random wireless sensor networks," in *INFOCOM*, 2007.