DEAMON: Energy-efficient sensor monitoring

Minho Shin, Patrick Tsang, David Kotz, Cory Cornelius
Institute for Security, Technology, and Society
Dartmouth College, Hanover, NH USA

***Abstract*—In people-centric opportunistic sensing, people offer their mobile nodes (such as smart phones) as platforms for collecting sensor data. A sensing application distributes sensing 'tasks,' which specify what sensor data to collect and under what conditions to report the data back to the application. To perform a task, mobile nodes may use on-board sensors, a body-area network of personal sensors, or sensors from neighboring nodes that volunteer to contribute their sensing resources. In all three cases, continuous sensor monitoring can drain a node's battery.**

We propose DEAMON (Distributed Energy-Aware MONitoring), an energy-efficient distributed algorithm for long-term sensor monitoring. Our approach assumes only that mobile nodes are tasked to report sensor data under conditions specified by a Boolean expression, and that a network of nearby sensor nodes contribute to monitoring subsets of the task's sensors. Our algorithm to select sensor nodes and to monitor the sensing condition conserves energy of all nodes by limiting sensing and communication operations. We evaluate DEAMON with a stochastic analysis and with simulation results, and show that it should significantly reduce energy consumption.

I. Introduction

Many have touted the opportunities available for *people-centric* sensing, in which mobile sensor devices are carried by people and provide information about people, typically about their activities and the surrounding physical and social environment. Whether the sensing is *opportunistic* or *participatory* [1], [2], the sensing application generates and distributes sensing "tasks," which specify what sensor data to collect and under what condition to report the data back to the application. Regardless of whether tasks are generated by Internet-based applications or by mobile applications running on personal devices, we assume that the tasks are ultimately distributed to one or more "mobile nodes" carried by people, who we call "carriers." The task-distribution mechanism is outside the scope of this paper.

To perform a task, mobile nodes may use on-board sensors, a body-area network of personal sensor nodes, or sensors from neighboring peers that volunteer to contribute their sensing resources. Eisenman et al. proposed this third form, and showed that this "sensor sharing" can improve data availability and data quality by allowing a node to overcome its limited set of sensors and to leverage neighbors that are better suited to provide the data [3]–[5].

Many sensing tasks continuously monitor sensors to decide when to collect and report data. Aggressive monitoring can quickly drain batteries, requiring frequent recharges and discouraging carriers from accepting tasks or sharing sensors.

We propose DEAMON (Distributed Energy-Aware MONitoring), an energy-efficient distributed algorithm for long-term sensor monitoring. Our approach assumes only that mobile nodes are tasked to report sensor data under conditions specified by a Boolean expression, and that a network of nearby sensor nodes contribute to monitoring subsets of the task's sensors. Our algorithm selects sensors– and sensor nodes– according to the energy cost of monitoring those sensors, and monitors only the low-energy subsets of the sensing condition needed to detect when the task should report sensor data.

This paper makes the following **contributions**.

- We formalize the problem of energy-efficient sensor monitoring, where some sensors are on another node.
- We propose DEAMON, an energy-efficient distributed algorithm that solves this problem.
- We evaluate DEAMON with a mathematical analysis and with simulation.

The results of our evaluation show that DEAMON significantly reduced the energy consumption of sensor monitoring, and that DEAMON scaled well with the complexity of the task and with the number of sensor nodes.

In the remainder of the paper we discuss related work (Section II), specify our system model (Section III), describe our algorithm (Section IV), derive a stochastic analysis (Section V), provide the results of our simulation (Section VI), discuss the results and opportunities for future work (Section VII), and conclude (Section VIII).

II. Related Work

Eisenman et al. [3], [4] introduced the concept of sensor sharing; their Quintet system supports "direct sensor sharing," in which a tasked node can discover, and obtain data from, suitable neighbor nodes. Quintet's sensor-selection algorithm chooses neighbors that can provide the highest data fidelity. Their evaluation focuses on the potential for sensor sharing to increase the probability that a task will obtain the desired data in time, or that it will increase the fidelity of the data; their work is orthogonal with our work in that they do not explicitly study the cost of sensing and communication, in particular, of sensor monitoring. In this paper, DEAMON treats direct sensor sharing as one form of distributed sensor monitoring; our algorithm and analysis focus on the energy cost of sensing and communication in sensor monitoring.

Our solution resembles the approach in SeeMon [6]. SeeMon proposed an energy-efficient context-monitoring framework. As a part of the solution, they proposed to monitor only essential sensors that determine the truth-values of Boolean conjunctive clauses. SeeMon uses a greedy Set-Cover algorithm to select those essential sensors. In this

paper, we propose a distributed sensor-monitoring algorithm to monitor a *general* Boolean expression that is split and assigned to multiple sensor nodes. We also provide a comprehensive mathematical analysis of the algorithm.

Many event-detection schemes have been proposed for multi-hop sensor networks. Existing solutions are for simple event descriptions, such as attribute-based events [7], threshold-based predicates [8]–[12], or pattern-matching events [13]. Detection of non-parametric complex events was proposed [14]; it requires data collection and data processing at centralized sites to learn the environment and detect unusual events. Our work instead aims to monitor parametric events described by generic Boolean expressions.

Our technique of selectively turning off some sensors is related to the "suppression technique" in sensor networks [15]–[18], which focuses on saving energy during data collection. Our technique, on the other hand, focuses on saving energy during condition monitoring. While both techniques save energy by avoiding unnecessary sensing, the suppression technique suppresses *redundant* data that can be inferred from other data, whereas ours suppresses *useless* data that does not help determine the status of the condition.

III. System Model

In this section, we describe our system model for sensing and distributed condition monitoring.

A. Sensing

We build DEAMON on a *task-based* sensing model [19], [20]. In this model, an *application* creates a *task* and distributes it to one or more *mobile nodes*, each of which is a smart phone or other personal device of a human *carrier*. The mobile nodes interpret the task's instructions, collect sensor data, and submit reports back to the application.

In a centralized system architecture, such as that in AnonySense [20], applications submit tasks through the Internet to a central task service, which then distributes the tasks to mobile nodes. In a peer-to-peer architecture, applications may run on the mobile nodes themselves; an application's tasks may be executed locally or may be distributed to other mobile nodes for execution. DEAMON is agnostic to either of these models; we simply assume that a task has arrived at a mobile node and should be executed there.

A task $T = (S, \mathcal{F}, r_R, t_B, t_E)$ defines what sensor data to report (set of sensor types S) under what condition (Boolean expression $\mathcal{F}$), how often to report (reporting rate r_R), and when to begin and end executing the task (timestamps t_B and t_E, respectively). $\mathcal{F}$ is a Boolean expression defined on S. Between time t_B and t_E, the mobile node monitors the sensing condition $\mathcal{F}$. While it holds true, the mobile node collects sensor readings in S at rate r_R and reports back to the application.

As mentioned above, mobile nodes may use on-board sensors, a body-area network of personal sensor nodes, or sensors from neighboring peers that volunteer to contribute their sensing resources (sensor sharing [3], [4]).

B. Sensing condition

DEAMON assumes that the sensing condition $\mathcal{F}$ is given in the conjunctive normal form (CNF), i.e., a conjunction of k clauses $\mathcal{C}_1, \ldots, \mathcal{C}_k$, and each clause $\mathcal{C}_i$ is a disjunction of l_i atoms $x_{i1}, \ldots, x_{il_i}$. In other words,

$$\mathcal{F} = \bigwedge_{i=1,\ldots,k} (x_{i1} \vee x_{i2} \vee \cdots \vee x_{il_i}). \tag{1}$$

Each atom x_{ij} takes as its value the output by some Boolean-valued function f_{ij} on a sensor reading as input.[1] For example, the following sensing condition

$$\mathcal{F}^* = \quad (speed > 10) \ \wedge \ (luminance \leq 5) \ \wedge \ \begin{pmatrix} humidity & \geq & 90 & \vee \\ temperature & < & 0 & \vee \\ temperature & > & 100 & \end{pmatrix} \tag{2}$$

has the form of $x_{11} \wedge x_{21} \wedge (x_{31} \vee x_{32} \vee x_{33})$, in which, for example, x_{32} is true *if and only if* the temperature is below zero. One can think of this $\mathcal{F}^*$ as a condition that tests whether a carrier is driving in the dark during a bad weather.

DEAMON supports arbitrary sensing conditions as any Boolean expression can be converted into CNF. We recognize that some Boolean expressions may convert to an exponentially long CNF. A longer expression is undesirable because it requires more computation for sensor assignment and more communication to monitor a large number of sub-formulas assigned to sensor nodes. In Section VII, we show how DEAMON can support DNF-conditions natively. Therefore, one can choose to convert the sensing condition into either the CNF or the DNF, whichever leads to a shorter expression.

Here we introduce some terminology. Given a sensing condition $\mathcal{F} = \bigwedge_{i=1}^{k} \mathcal{C}_i$, we say that C *is a clause in* $\mathcal{F}$ if $C \in \{\mathcal{C}_i\}$, and that x *is an atom in* C if $C = \cdots \vee x \vee \cdots$. We write $A(\mathcal{F})$ to denote the atom-set of $\mathcal{F}$, which is the set of all atoms that appear in $\mathcal{F}$. We say that a clause C *is a sub-clause of* $\mathcal{F}$ if all atoms in C appear in a clause of $\mathcal{F}$.

C. Distributed condition monitoring

Consider a node m, which is equipped with a sensor set S_m and which needs to perform a task $T = (S, \mathcal{F}, r_R, t_B, t_E)$. If some of the required sensors are missing ($S \not\subseteq S_m$), then node m needs to obtain sensor data from outboard sensor nodes or from neighboring volunteers. Even if m has all the required sensors, it may choose to borrow some of the required sensors from neighboring nodes to improve data quality [3] or to save its energy. DEAMON allows a *master* node m to select *helper* nodes h, with the goal of minimizing energy consumption when monitoring the sensing condition.

To identify helpers, the master m broadcasts its task T through a short-range radio (e.g., Bluetooth) and waits for responses from nearby nodes. Each neighboring node n considers the task T, its own sensor set S_n, and its sensor-sharing

[1] In fact, DEAMON supports the case when an atom's value depends on multiple sensor readings. For clarity, however, we assume that f_{ij} is a single-variable function.

policy. It becomes a helper when it replies to the master, offering a set of sensors $O_n \subseteq S_n \cap S$. With the most generous policy, n may offer every sensor it has, i.e., $O_n = S_n \cap S$. Some carriers, however, may have more restrictions on sharing their sensors with others. For example, some carriers may not want to share their GPS readings because of privacy concerns, while some may not want to share video recordings because of the energy consumption. A carrier's sensor-sharing policy and its security implication is outside the scope of this paper.

Given the offers from the helpers, the master must perform *condition assignment*; the master breaks the sensing condition $\mathcal{F}$ into sub-clauses, each assigned to one of the helpers, in such a way that the master can monitor $\mathcal{F}$ if each helper duly monitors its assigned sub-clauses. DEAMON's *condition-monitoring* algorithm allows the master to, with help from the helpers, detect when the sensing condition $\mathcal{F}$ becomes true. While the condition remains true, the master collects readings from sensors in S at rate r_R, and reports to the application.

IV. DEAMON

We first formalize the energy-aware sensor monitoring problem and then describe our solution.

Let $\mathcal{U}$ be the universe of all sensor types supported in the system. Let m be the master node and H be the set of helpers. Since the master m also contributes its own sensors, $m \in H$. For each helper $h \in H$, let $O_h \subseteq \mathcal{U}$ be h's offer-set and assume that the union of offer-sets covers the required sensors. Let $c^r_{h,s}$ be h's sampling cost on sensor $s \in O_h$ and c^t_h be h's transmission cost per bit. We call $\mathbf{O}_H = \{O_h : h \in H\}$ the sensor profile and $\mathbf{C}_H = \{(c^r_{h,s}, c^t_h) : h \in H\}$ the cost profile.

Definition 1 (Energy-aware sensor monitoring): Given $(\mathcal{F}, H, \mathbf{O}_H, \mathbf{C}_H)$ as input, the *energy-aware sensor monitoring* problem is to detect when $\mathcal{F}$ holds true with the help of helpers in H characterized by $\mathbf{O}_H$ and $\mathbf{C}_H$. The goal is to minimize the *average energy consumption rate*

$$C_{rate} = \sum_{h \in H} \left\{ \left(\sum_{s \in O_h} \tau_{h,s} \cdot p_{h,s} \cdot c^r_{h,s} \right) + w_h \cdot c^t_h \right\} \quad (3)$$

where $\tau_{h,s}$ is h's sampling rate on sensor s, $p_{h,s}$ is the time proportion that h monitors sensor s and w_h is the average transmission bandwidth of h. Therefore, one can minimize C_{rate} by minimizing $p_{h,s}$ and w_h.

DEAMON aims to solve the energy-aware sensor monitoring problem by solving two sub-problems. The *condition-assignment* problem is to break the sensing condition into a set of sub-clauses, each assigned to one of the helpers. Given a condition assignment, the *condition-monitoring* problem is to cost-effectively monitor the sensing condition by controlling the helper's monitoring activities on assigned sub-clauses.

A. DEAMON's condition assignment

We present our DEAMON-ASSIGN algorithm to solve the condition-assignment problem. On input $(\mathcal{F}, H, \mathbf{O}_H, \mathbf{C}_H)$, DEAMON-ASSIGN outputs an assignment $\mathbf{A} = \{F_h : h \in H\}$ where F_h is a set of sub-clauses to be assigned to h.

Elaborating on the example of $\mathcal{F}^* = x_{11} \wedge x_{21} \wedge (x_{31} \vee x_{32} \vee x_{33})$ in Eq. 2, if there are two helpers h_1 and h_2 that have an offer-set of $O_{h_1} = \{speed, humidity, temperature\}$ and $O_{h_2} = \{speed, luminance, temperature\}$ respectively, then one possible assignment is $\mathbf{A} = \{F_{h_1}, F_{h_2}\}$, where $F_{h_1} = \{x_{11}, x_{31}\}$ and $F_{h_2} = \{x_{21}, x_{32} \vee x_{33}\}$. Assigned with F_{h_2}, helper h_2 detects when it is too dark (x_{21}) or when the temperature is extreme ($x_{32} \vee x_{33}$).

DEAMON-ASSIGN assigns the atoms in $A(\mathcal{F})$ to the helpers such that the expected monitoring cost in Eq. 3 $E[C_{rate}]$ is minimized. However, it is expensive to directly find the minimum $E[C_{rate}]$; one must compute $p_{h,s}$ for all h for all combinations of possible assignments, i.e., solving an exponential number of linear systems (see Section V). Therefore, we estimate $E[C_{rate}]$ with a simple assumption that every sensor is monitored all the time ($p_{h,s} = 1$) and helpers report whenever assigned atoms change their values, i.e.,

$$\hat{E}[C_{rate}] = \sum_{h \in H} \sum_{x \in A(F_h)} Cost_h(x) \quad (4)$$

where

$$Cost_h(x) = \sum_{s \in \sigma(x)} \left(\tau_{h,s} \cdot c^r_{h,s} \right) + \lambda_x \cdot L_h \cdot c^t_h. \quad (5)$$

In Eq. 5, $\sigma(x)$ denotes the sensor-set that determines the value of atom x, λ_x is the expected change rate of x and L_h is the length of a reporting message that h transmits. DEAMON-ASSIGN aims to minimize $\hat{E}[C_{rate}]$.

One can show that the assignment problem above is a variant of the *weighted set-cover problem*, which is NP-complete [21]. Since the weighted set-cover problem is a special case of our assignment problem (when $Cost_h(x) = g(h)$ for some real function g), the condition-assignment problem is NP-hard. The greedy algorithm for the weighted set-cover problem is asymptotically close to the best possible approximation algorithm [22]. Therefore, we use a similar greedy algorithm in DEAMON-ASSIGN.

Algorithm 1 describes the DEAMON-ASSIGN algorithm. Let C be the current "cover", i.e., the set of atoms that have already been assigned to some helper, and let X_h be the set of atoms that h can monitor with O_h. Then, $X_h \setminus C$ denotes yet uncovered atoms that h can cover. The algorithm chooses a helper that has the smallest average per-atom cost for monitoring atoms in $X_h \setminus C$. This is a reasonable heuristic because, if we add as small a newly-added cost as possible for covering each atom, we will eventually minimize the total cost of covering all atoms. Then we assign to the chosen helper those newly-covered atoms C_h, converted to sub-clauses by the COLLAPSE algorithm. COLLAPSE, which is not explicitly defined due to space constraints, combines atoms that belong to the same clause into one disjunctive formula. For example, if $C_h = \{x_{12}, x_{32}, x_{33}\}$, then $F_h = \{x_{12}, (x_{32} \vee x_{33})\}$. This compression reduces unnecessary communication between the helpers and the master when the atoms change but the corresponding sub-clause does not change. The algorithm iterates this process until all atoms in $A(\mathcal{F})$ have been assigned.

Algorithm 1 DEAMON-ASSIGN($\mathcal{F}, H, \mathbf{O}_H, \mathbf{C}_H$)

Notation:
$A(\mathcal{F}) \equiv$ the atom-set of $\mathcal{F}$
$X_h \equiv$ set of atoms that sensors in O_h can determine

1: $C := \emptyset$; $H_C := H$
2: **while** $A(\mathcal{F}) \not\subseteq C$ **do**
3: **if** $H_C = \emptyset$ **then**
4: exit("set-cover failed")
5: **end if**
6: $h = \text{argmin}_{h \in H_C} \dfrac{\sum_{x \in X_h \setminus C} Cost_h(x)}{|X_h \setminus C|}$
7: $C_h := X_h \setminus C$
8: $F_h :=$COLLAPSE$(C_h, \mathcal{F})$
9: $H_C := H_C \setminus \{h\}$
10: $C := C \cup C_h$
11: **end while**
12: **output** $\mathbf{A} = \{F_h : h \in H\}$

B. Basic condition monitoring

Consider the following simple but effective condition-monitoring algorithm ("BASIC"). Each helper monitors its assigned sub-clauses by monitoring all sensors in those sub-clauses, notifying the master whenever any sub-clause's truth value changes. This solution, however, can waste the helper's energy when not all sensor readings are needed to determine the validity of the sub-clause or the sensing condition.

C. DEAMON's condition monitoring

DEAMON saves energy by monitoring only those sensors necessary to determine the truth of the expression. Other sensors are not monitored until needed. We first describe our approach and data structure, and then explain the algorithm.

Approach.

Boolean expression $\mathcal{F}$ has two states: $\mathbf{T}$ for true and $\mathbf{F}$ for false. Our goal is to detect transitions between $\mathbf{T}$ and $\mathbf{F}$. One can detect such a transition by monitoring all the atoms, all the time (as BASIC algorithm does). Instead, our algorithm only monitors the smallest set of atoms that is sufficient to determine the current state of $\mathcal{F}$.

For example, consider the following Boolean expression:

$$\mathcal{F} = (x_1 \vee x_2 \vee x_3) \wedge (x_4 \vee x_5 \vee x_6). \quad (6)$$

For $\mathcal{F}$ to be true, it is sufficient that at least one atom is true in each clause. For example, $\mathcal{F}$ = true if $x_1 = x_5$ = true regardless of other atoms. In this case, we only monitor x_1 and x_5; and we say that we are in state $T_{(1,5)}$. For $\mathcal{F}$ to be false, it is sufficient that there exists a clause whose atoms are all false (e.g., $\mathcal{F}$ = false if $x_4 = x_5 = x_6$ = false). Therefore, we only monitor x_4, x_5, and x_6 in this case; and we are in state F_2. In general, given $\mathcal{F} = C_1 \wedge \cdots \wedge C_k$, there are $\prod_{i=1}^{k} |C_i|$ true states, each denoted by $T_{(I_1, I_2, \ldots, I_k)}$ where I_i is the index of an atom $x_{I_i} \in C_i$ such that x_{I_i} = true and we monitor only those atoms. There are k false states $F_1, F_2, \ldots, F_k$ where F_i denotes when we monitor clause C_i whose atoms are all false.

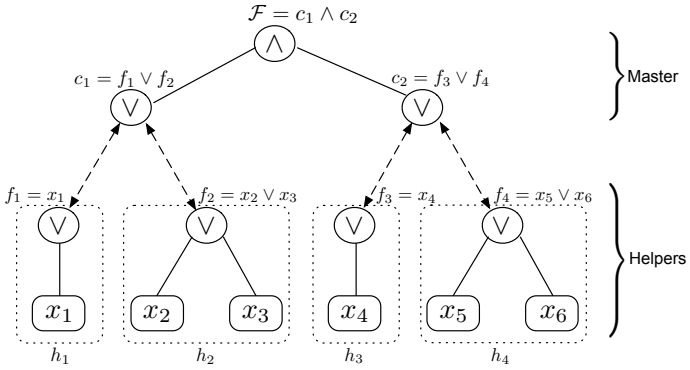


Fig. 1. A sensing condition $\mathcal{F} = (x_1 \vee x_2 \vee x_3) \wedge (x_4 \vee x_5 \vee x_6)$ is represented by an evaluation tree given the assignment $\mathbf{A} = \{\{x_1\}, \{x_2 \vee x_3\}, \{x_4\}, \{x_5 \vee x_6\}\}$ for helpers h_1, h_2, h_3, and h_4, respectively. Each level is called Root, Clause, Sub-clause, and Atom levels, from top to bottom.

Our algorithm is always in one of the states defined above. When we are not in the current state any more, i.e., any of x_{I_i} becomes false when we are in a T state, or any atom in clause C_i becomes true when we are in F state, then the algorithm examines further and determines the new state. For example, suppose we were in state $T_{(1,5)}$ where we are only monitoring x_1 and x_5. When x_1 becomes false, we are not in $T_{(1,5)}$. Therefore, we read x_2 and x_3. If x_2 = true, we are in $T_{(2,5)}$ and $\mathcal{F}$ = true. If $x_2 = x_3$ = false, we are in F_1.

Data structure.

Assume that we are given Eq. 6 and helpers h_1, h_2, h_3, h_4 are assigned $\{x_1\}, \{x_2 \vee x_3\}, \{x_4\}$ $\{x_5 \vee x_6\}$, respectively. Then, we can construct an *evaluation-tree* $Etree(\mathcal{F})$ as shown in Figure 1. Each internal node $\otimes$ or $\oslash$ represents a disjunctive or conjunctive connection of its children. From top to bottom, each level represents sensing condition, clause, sub-clause, and atoms. The upper part of the data structure is maintained by the master node while the lower part is split and assigned to helpers. The dashed lines between sub-clause and clause denotes wireless communication between the master and the helpers. We sort the children of each node in an increasing order of their estimated monitoring cost based on Eq. 5. For example, C_1 appears earlier than C_2 because $Cost_{h_1}(x_1) + Cost_{h_2}(x_2) + Cost_{h_2}(x_3)$ is smaller than $Cost_{h_3}(x_4) + Cost_{h_4}(x_5) + Cost_{h_4}(x_6)$. Our algorithm monitors lower-indexed atoms whenever possible.

Algorithm.

Algorithms 2 and 3 describe the DEAMON-MONITOR algorithm, which defines the system's reaction to two events: an atom being monitored becomes true or false. The initial state is F_1, i.e., we monitor all the atoms in the first clause. Because of this initial status, the first event to occur is "x^* becomes true" on some helper h; h notifies master m of this event. Then, m stops monitoring the clause that contains x^* (line 3) and examines other clauses (line 4). If it finds a clause whose atoms are all false (line 5) the master starts monitoring

Algorithm 2 DEAMON-MONITOR

EVENT: x^* becomes true in h

1: $h \to m$: (notify, x^*, true)
2: $X := \{x^*\}$
3: MONITOR(CLAUSE(x^*), off)
4: **for each** clause $c \in \mathcal{F} \setminus$ CLAUSE(x^*) in the ascending order of monitoring energy **do**
5: $x =$ FINDTRUEATOM(c)
6: **if** $x = \emptyset$ **then**
7: MONITOR(c, on)
8: **return** false
9: **else**
10: $X = X \cup \{x^*\}$
11: **end if**
12: **end for**
13: MONITOR(X, on)
14: **return** true

EVENT: x^* becomes false in h

15: $h : x =$ FINDTRUEATOM(SUBCLAUSE(x^*))
16: **if** $x \neq \emptyset$ **then**
17: h : on(x), off(x^*)
18: **return** true
19: **end if**
20: $h \to m$: (notify, x^*, false)
21: $x =$ FINDTRUEATOM(CLAUSE(x^*))
22: **if** $x \neq \emptyset$ **then**
23: MONITOR(x, on)
24: MONITOR(x^*, off)
25: **return** true
26: **end if**
27: MONITOR(CLAUSE(x^*), on)
28: **return** false

Algorithm 3 Sub-routines for DEAMON-MONITOR

1: **procedure** MONITOR(S, val)
2: **for each** $x \in S$ **do**
3: h: the node that hosts sensor x
4: $m \to h : (\mathsf{mon}, x, val)$
5: **end for**
6: **end procedure**
7: **procedure** FINDTRUEATOM(c)
8: **for each** $h \in$ HELPERS(c) **do**
9: $m \to h$: (query, "who is true?")
10: **if** $h \to m : (\mathsf{notify}, x, \mathsf{true})$ **then**
11: **return** x
12: **end if**
13: **end for**
14: **return** $\emptyset$
15: **end procedure**

that clause (lines 6–8), and concludes that $\mathcal{F} =$ false (entering state F_j). Otherwise, it monitors atoms in all true clauses (entering $T_{(\dots)}$) and concludes that $\mathcal{F} =$ true (lines 13–14).

When "x^* becomes false", the helper first checks if there is another true-valued atom in the same sub-clause, i.e., under the same helper. If so, h monitors the new atom instead of x^* (entering $T_{(\dots)}$) and need not report to m. Otherwise, the master gets a notification (line 20) and seeks another true-valued atom within the same clause (line 21). If so, it starts monitoring the new atom (entering $T_{(\dots)}$) and $\mathcal{F} =$ true (lines 22–26). Otherwise, i.e., all atoms in the clause are false, then m monitors all the atoms in the clause (entering F_i) and concludes that $\mathcal{F} =$ false.

In the Algorithms 2 and 3, CLAUSE(x) and SUBCLAUSE(x) return the clause and the sub-clause that contains atom x, respectively. MONITOR() tells all relevant helpers to start or stop monitoring atoms within S. FINDTRUEATOM() finds a true-valued atom within given clause or sub-clause by sending query message to helpers. HELPERS(c) returns the set of helpers under the clause or the sub-clause.

V. COST ANALYSIS

To analyze the energy cost of the above algorithms, we first introduce a stochastic model for the dynamics of the environment as a basis of the cost analysis.

Our model assumes that the environment changes over time but there exist invariants that describe the change. Let x be an atom, such as "$x =$ (*temperature* < 65)." As the environment changes over time, the value of x alternates between false and true. Let us denote the ith duration when $x =$ false by Y_i and the ith duration when $x =$ true by Z_i. We assume that the sequence of random variables $\{Y_i\}$ is independent and identically distributed, and $\{Z_i\}$ is likewise. Then, the value of x follows a stochastic process called an *alternating renewal process* [23]. The following analysis is independent of specific distributions for $\{Y_i\}$ and $\{Z_i\}$, but we assume exponential distributions in our simulations.

Let their expected values be $E[Y_i] = 1/\lambda_x^0$ and $E[Z_i] = 1/\lambda_x^1$ where λ_x^0 is the average rate at which x changes from false to true, and λ_x^1 is the average rate at which x changes from true to false. By averaging $E[Y_i]$ and $E[Z_i]$, we obtain x's average change rate $\lambda_x = \frac{2\lambda_x^0\lambda_x^1}{\lambda_x^0+\lambda_x^1}$. We also define $p_x = \frac{1/\lambda_x^0}{1/\lambda_x^0+1/\lambda_x^1}$ as the long-run probability that x is false. We assume that (p_x, λ_x) characterizes the dynamics of x and, for simplicity, remains the same during the sensing task.

A. Analysis of BASIC

The average cost rate of h monitoring $f \in F_h$ is

$$Cost_h^{\mathsf{BASIC}} = \sum_{f \in F_h} \left\{ \sum_{s \in \sigma(f)} \tau_{h,s} \cdot c_{h,s}^r + \lambda_f \cdot L \cdot c_h^t \right\} \quad (7)$$

where λ_f is the expected change rate of sub-clause f. Let us now derive λ_f assuming that $\{Y_i\}$ and $\{Z_i\}$ follow exponential distributions (other distributions are left to interested readers). Let $f = x_1 \vee \cdots \vee x_k$. Then, $\lambda_f^0 = \sum_i \lambda_i^0$, the sum of rates at which an atom becomes true. Also, $\lambda_f^1 = \sum_i P[\text{only } x_i =$

$\mathsf{true}] \cdot \lambda_i^1 = \sum_i (1-p_{x_i})\Big(\prod_{j\neq i} p_{x_j}\Big)\lambda_i^1$. Since $p_f = \prod_i p_{x_i}$, $\lambda_f = \frac{2\lambda_f^0\lambda_f^1}{\lambda_f^0+\lambda_f^1}$, $\lambda_x^0 = \frac{\lambda_x}{2p_x}$, and $\lambda_x^1 = \frac{\lambda_x}{2(1-p_x)}$, we get

$$\lambda_f = \frac{1}{(1+\prod_i p_i)} \sum_{i=1}^{k} \Big(\lambda_{x_i} \prod_{\substack{j=1 \\ j\neq i}}^{k} p_{x_j}\Big). \tag{8}$$

Therefore, the total energy-cost rate of BASIC is

$$Cost^{\mathsf{BASIC}} = \sum_{h\in H} Cost_h^{\mathsf{BASIC}}. \tag{9}$$

B. Markov model of DEAMON

Recall that DEAMON is always in a F_i or a $T_{(I_1,\ldots,I_k)}$ state. In this section, we analyze the algorithm as a *continuous-time Markov process*; the current state depends on the state at a prior time t but not on the history up to time t. More importantly, this Markov process has a stationary distribution since it is finite, irreducible (any state can transit to any other state), and ergodic (any state can be revisited within a finite time). We first derive the transition matrix $\mathbf{Q}$ (transition rates between states) and compute the stationary distribution vector $\boldsymbol{\pi}$ (the probabilities of being in each state).

Transition $F_i \to F_j$**.** In lines 5–8 in Algorithm 2 we transit from state F_i to state $F_j (j \neq i)$ when one atom in clause C_i turns true *and* we find a clause $C_j (j \neq i)$ whose atoms are all false. The transition rate is the rate at which we leave F_i multiplied by the probability that we enter F_j. Since the rate of false-valued C_i becoming true is sum of the rates of an atom becoming true, and all the clauses before we reach C_j had at least one true-valued atom, we get

$$q_{(F_i,F_j)} = \Big(\sum_{x\in C_i} \lambda_x^0\Big)\Big(\prod_{x\in C_j} p_x\Big) \prod_{\substack{l=1 \\ l\neq i}}^{j-1} \Big(1 - \prod_{x\in C_l} p_x\Big). \tag{10}$$

Transition $F_i \to T_{(I_1,\ldots,I_k)}$**.** One atom in C_i became true, specifically, x_{I_i}. We discover at line 5 that x_{I_j} is true (with probability of $1-p_{x_{I_j}}$), and that atoms before x_{I_j} in the same clause were false (with probability of $p_{x_{Il}}$). Therefore,

$$q_{(F_i,T_{(\ldots)})} = \lambda_{x_{I_i}}^0 \prod_{j\neq i} \Big\{(1-p_{x_{I_j}}) \prod_{\substack{x_l\in C_j \\ l<I_j}} p_{x_l}\Big\}. \tag{11}$$

Transition $T_{(I_1,\ldots,I_k)} \to F_i$**.** This transition happens when atom x_{I_i} in clause C_i becomes false while all other atoms in that clause remain false, thus $C_i = \mathsf{false}$ (line 27). Therefore, the transition rate is the rate at which the atom becomes false multiplied by the probability that other atoms are also false:

$$q_{(T_{(\ldots)},F_i)} = \lambda_{x_{I_i}}^1 \prod_{\substack{x_l\in C_i \\ l<I_i}} p_{x_l}. \tag{12}$$

Transition $T_{(I_1,\ldots,I_k)} \to T_{(J_1,\ldots,J_k)}$**.** The transition from T-state to T-state happens when an atom turns false but we find another true-valued atom within the same clause. There are two cases; either $x_{I_{i^*}}$ and $x_{J_{i^*}}$ belong to the same sub-clause (i.e., same helper) or not.

When they belong to the same sub-clause f (line 16), the transition rate is the rate at which the atom $x_{I_{i^*}}$ becomes false multiplied by the probability that atom $x_{J_{i^*}}$ is found true and preceding atoms within the sub-clause remain false. Thus,

$$q_{(T_{(I)},T_{(J)})} = \lambda_{x_{I_{i^*}}}^1 (1-p_{x_{J_{i^*}}}) \prod_{\substack{x_l\in f \\ l\neq I_{i^*} \\ l<J_{i^*}}} p_{x_l}. \tag{13}$$

When they belong to different sub-clauses but the same clause (line 22), we find that all other atoms in the same sub-clause are false and that other atoms in the sub-clauses observed before we reach $x_{J_{i^*}}$ are false. Therefore,

$$q_{(T_{(I)},T_{(J)})} = \lambda_{x_{I_{i^*}}}^1 (1-p_{x_{J_{i^*}}}) \prod_{\substack{x_l\in f \\ l\neq I_{i^*}}} p_{x_l} \prod_{\substack{x_n\notin f \\ n<J_{i^*}}} p_{x_n}. \tag{14}$$

We can compute $\boldsymbol{\pi} = \{\pi_s : s \in \Sigma\}$ by solving a linear system obtained by equating the total outgoing rates with the total incoming rates of each state, subject to $\sum_s \pi_s = 1$.

C. DEAMON: Reading cost

Given a stationary distribution $\boldsymbol{\pi}$, we can compute the average reading cost of DEAMON as follows. When we are in state F_i, all atoms in clause C_i are being monitored. That is, the cost of reading in F_i is $\sum_{s\in\sigma(C_i)} \tau_{h,s} \cdot c_{h,s}^r$ where $\sigma(C_i)$ is the set of sensors that determines the value of C_i. When in state $T_{(I_1,\ldots,I_k)}$, the reading cost is $\sum_{i=1}^k \tau_{h,\sigma(x_{I_i})} \cdot c_{h,\sigma(x_{I_i})}^r$. Let $\mathcal{I}$ denote the set of all possible combinations of k atom indexes, each chosen from each clause. Then, the total cost rate for reading sensors of DEAMON-MONITOR is

$$\begin{aligned} Cost_{DEAMON}^r = &\sum_{i=1}^{k} \Big\{\pi_{F_i} \cdot \sum_{s\in\sigma(C_i)} \tau_{h,s} \cdot c_{h,s}^r\Big\} \\ &+ \sum_{I\in\mathcal{I}} \Big\{\pi_{T_I} \cdot \sum_{i=1}^{k} \tau_{h,\sigma(x_{I_i})} \cdot c_{h,\sigma(x_{I_i})}^r\Big\} \end{aligned} \tag{15}$$

D. Communication cost of DEAMON

In this section, we compute the energy cost for communication between the master and helpers: the master sends query messages to learn current atom values (line 9 in Algorithm 3); helpers send notification messages to let the master know atom values (lines 1 and 20 in Algorithm 2 and line 10 in Algorithm 3); and the master sends set-monitor messages to control monitoring status of atoms (when MONITOR() is executed in Algorithm 3). Notice that messages of all three types are in a form of (id, val) where id refers to a sub-clause or an atom, and val is a one-bit data. Therefore, we assume that all messages have the same length of L.

Because of the formulas' complexity and limited space, we only present the communication cost of transition from F_i to F_j, which has the most complex formula among transitions. The same method applies to the other transitions.

Communication for $F_i \to F_j$. The number of query messages is $|C_j|$, to verify that all atoms in C_j are false, plus the number of probes on other clauses which stopped in

the middle when we find a true-valued atom. Let E_{C_l} denote the expected number of probes on a clause C_l until we find an atom that is true, conditioned that such an atom exists in the clause. Then,

$$E_{C_l} = \sum_{j=1}^{|C_l|} j \cdot (1 - p_{x_j}) \prod_{n=1}^{j-1} p_{x_n}. \quad (16)$$

Then, the average number of queries is $N_q = (|C_j| + \sum_{l=1}^{j-1} E_{C_l})$, the average number of notifications is $N_n = N_q + 1$ (the extra one is for notifying the change of an atom in clause C_i), and the average number of set-monitor messages is $N_m = |C_i| + |C_j|$. Therefore, the total expected energy cost for communication during the transit from F_i to F_j is

$$\left\{ (N_q + N_m) \cdot c_m^t \cdot L + N_n \cdot \hat{c}^t \cdot L \right\} \cdot q_{(F_i, F_j)} \quad (17)$$

where $\hat{c}^t = \sum_{h \neq m} c_h^t / (|H| - 1)$ is the average one-bit transmission cost of all helpers other than the master. Likewise, we can compute the communication costs for other types of transitions; we add all such costs to get the total communication cost of DEAMON-MONITOR.

VI. Evaluation

For verification, we compare our analytical predictions with simulation results. We developed a custom simulator, which emulated only abstract message passing and ignored realistic network conditions. However, all the energy consumption regarding network activities are captured by the simulator. We chose parameter values as shown in Table I.

TABLE I
Simulation parameters

Parameters	Notation	Value
Atom change distribution		exponential distribution
Atom change interval	$1/\lambda_x$	60 $\sim$ 100 seconds
Atom false probability	p_x	0.1 $\sim$ 0.9
Atom reading cost	C_h^r	0.1 $\sim$ 0.9 units
Sampling rate	τ	2 times per second
Per-bit Transmission cost	C_h^t	0.1 $\sim$ 0.9 units
Message length	L	20 bytes (Bluetooth)

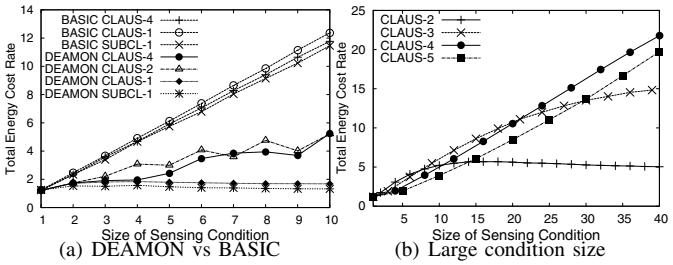
We varied the number of atoms in the following two formats:

$$\text{CLAUS-}K = \wedge_i (x_{i1} \vee x_{i2} \vee \cdots \vee x_{iK}) \quad (18)$$

$$\text{SUBCL-}K = \vee_i (x_{i1} \vee x_{i2} \vee \cdots \vee x_{iK}) \quad (19)$$

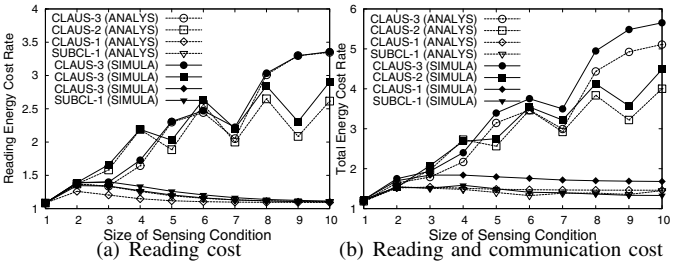
where SUBCL-K has one column and each helper is assigned K atoms. Note that CLAUS-1 is a simple conjunctive form and SUBCL-1 is a simple disjunctive form. The number of helper nodes was equal to the number of subclauses. If not stated otherwise (e.g., SUBCL-K), we assume that each subclause had only one atom.

For various combinations of parameters, we ran the monitoring algorithms for enough time to stabilize the algorithm. We report the average energy-cost per second from 20 repetitions. The variance was small so we omit it in the figures.



(a) DEAMON vs BASIC (b) Large condition size

Fig. 2. Energy cost by condition size



(a) Reading cost (b) Reading and communication cost

Fig. 3. Analysis vs Simulations

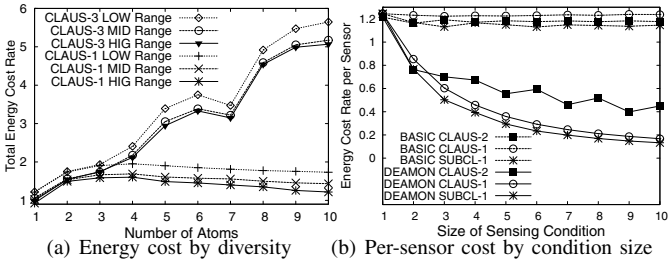
A. Simulation results

Overall. In Figure 2(a), we compare the energy cost of DEAMON and BASIC. The figure shows that DEAMON uses much less energy than BASIC in all types of conditions. DEAMON has nearly constant energy cost for simple conjunctive or disjunctive Boolean expressions (SUBCL-1, CLAUS-1). We omit results for SUBCL-2, 3, ... because their costs were almost identical to SUBCL-1. Thus, given a sensing condition, the assignment size to each helper and the number of helpers has little effect on the total cost. However, the growing number of clauses with multiple atoms increases the total cost sub-linearly. The fluctuations of CLAUS-2 and CLAUS-4 arise because the addition of atoms breaks the regularity of the formula every 2 or 4 atoms, respectively.

Figure 2(b) shows results with large conditions (up to 40 atoms). Fluctuations due to irregularity are suppressed for clarity. The figure shows that the energy cost converges to a limit when the clause has 2 or 3 atoms. In fact, we can mathematically show that it also converges with larger clauses by assuming that atoms become false with the same probability. By this assumption, the probability of the formula being false converges to 1 and, therefore, the energy cost converges to that of monitoring one clause. The convergence is slower for larger clauses.

Analysis vs. Simulation. We compare our analysis with simulation results in Figure 3(a) (only for sensor-reading cost) and Figure 3(b) (for total cost). The figures show that our analysis matches well with simulation results in various situations, although the fit is better when considering only sensor-reading cost. The discrepancy results from our estimation, rather than exact computation, of the communication costs (needed for simplicity; see Sections V-C and V-D).

Diversity. DEAMON cleverly monitors cheap-to-monitor atoms as much as possible and, we expect that we save more energy when the cost profile is diverse among devices. In Figure 4(a), we compare energy cost with different diversities;



(a) Energy cost by diversity (b) Per-sensor cost by condition size

Fig. 4. Diversity and per-sensor energy cost

LOW when devices have similar cost profile (between 0.4 and 0.6) and HIGH when cost profile spans large (between 0.1 and 0.9). The figure shows that DEAMON saves more energy as diversity increases; the biggest improvement was from low to mid diversities.

Cooperation effect. Finally, we conclude that DEAMON is achieving its fundamental goal of saving energy for each helper node, which may help to encourage sensor sharing. Specifically, Figure 4(b) shows that BASIC costs each helper a fixed amount of energy per second regardless of how many helpers are involved, but DEAMON decreases per-sensor energy cost as more helpers join the monitoring, thus decreasing the per-helper energy cost.

In summary,

- our stochastic analysis matches with simulation results,
- DEAMON significantly reduces the energy cost of monitoring a sensing condition and scales well,
- DEAMON reduces per-helper cost as collaboration grows,
- DEAMON benefits from the diversity of devices.

VII. Discussion and future work

In this section we discuss several issues related to our design and some opportunities for future work.

Model for atom dynamics. Our analysis and simulations assumed that an atom value changes with an exponential distribution. In fact, any other distribution can be assumed for similar results. However, we assumed that the probability p_x and change rate λ_x remains the same for long enough so that the node can estimate p_x and λ_x. The node can estimate p_x by estimating the density function of the sensor value [24], [25]. Likewise, the node can also estimate the distribution of interchange interval and then derive the average change rate [26], [27]. As nodes keep track of up-to-date parameters, DEAMON can adapt to the changing environment. As future work, we plan to evaluate DEAMON using a real sensor dataset, such as Intel Lab Data [28].

Interaction between assignment and monitoring. As described in Section IV-A, the DEAMON-ASSIGN algorithm estimates the expected energy consumption with an assumption that no energy optimization is performed. This simplification is due to the exponential number of combinations to consider for an exact computation. The master node could periodically poll the helpers to report their actual energy cost with DEAMON-MONITOR and reassign sensors based on the updated energy-consumption profile.

Sensor assignment for fairness. DEAMON-ASSIGN tries to minimize the total energy consumption of helpers in a greedy manner. Consequently, some helpers can contribute a lot more than other helpers. For fairness, we can redesign the assignment algorithm to minimize the difference between the least-contributing and the most-contributing helpers. This forms a *balanced* set-cover problem. One possible approximation algorithm would (at each iteration) choose the sensor that induces the least difference between max/min contributions.

DNF support. Because of duality between CNF and DNF, DEAMON can easily support DNF conditions; note that

$$(\text{CNF-}\mathcal{F} = \mathsf{true}) \text{ if } (\forall C \in \mathcal{F},\ \exists a \in C \mid a = \mathsf{true})$$

and also

$$(\text{DNF-}\mathcal{F} = \mathsf{false}) \text{ if } (\forall C \in \mathcal{F}, \exists a \in C \mid a = \mathsf{false})$$

where C is a clause and a is an atom. Since DEAMON (for CNF) uses the first inference to detect when CNF-$\mathcal{F}$ becomes true, we can make the algorithm detect when DNF-$\mathcal{F}$ becomes false by swapping true with false throughout Algorithms 2 and 3 (and changing the name FindTrueAtom to FindFalseAtom). Similarly, we can update our analysis by replacing p_x with $(1 - p_x)$ and swapping λ_x^0 and λ_x^1.

Multi-hop communication. Although we considered one-hop communication between the master and the helpers, DEAMON can apply to multi-hop networks to monitor a large area. For example, the master negotiates with a d-hop neighborhood, assigns sub-clauses to helpers, and performs the algorithms proposed in this paper. Because multi-hop communication can increase delay and communication cost (of forwarding nodes), the assignment algorithm need to account for the extra transmissions and prefer closer nodes.

Fidelity vs. Energy. Inherently, there is a trade-off between energy-efficiency and data quality. Eisenmann [3] proposed a fidelity metric for choosing sensors with better quality. We can consider both fidelity and energy by parametrizing the priority between them. For example, given a fidelity-energy weight $\alpha \in [0, 1]$, we can use a new cost function $\alpha Cost_h(x) + (1 - \alpha)1/Fid_h(s)$ in Algorithm 1 where $Fid_h(s)$ is a fidelity score of sensor s.

Dynamic assignment and multiple sensing-condition monitoring. Context-aware applications in a sensor-rich environment [29] will require mobile nodes to monitor multiple contexts [6] and take advantage of surrounding sensor nodes when they dynamically become available. DEAMON can be extended to support dynamic assignment and multiple sensing-condition monitoring. Based on fidelity and energy policy, the device can dynamically re-assign monitoring tasks to volunteering sensor nodes and optimize across multiple contexts.

VIII. Summary

In this paper, we describe an energy-efficient method for monitoring sensing conditions in a distributed setting. We formalize the energy-aware sensor-monitoring problem and propose DEAMON, a distributed energy-efficient sensor-monitoring algorithm that saves energy by tactically distributing the sensing condition to helpers and detecting the collection-triggering events. We evaluated our scheme with

mathematical analysis and simulations. Our results show that our analysis matches with the simulation results and that DEAMON should significantly reduce monitoring cost.

ACKNOWLEDGMENTS

This research program is a part of the Institute for Security, Technology, and Society (ISTS) at Dartmouth College, supported by Grants 60NANB6D6130 awarded by the U.S. Department of Commerce and 2006-CS-001-000001 awarded by the U.S. Department of Homeland Security. Points of view or opinions in this document are those of the authors and do not represent the official position or policies of the sponsors. We thank Dan Peebles and the other members of the Metrosense team at Dartmouth for their helpful comments.

REFERENCES

[1] J. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. B. Srivastava, "Participatory sensing," in *World Sensor Web Workshop (at Sensys)*, Oct. 2006. Available online: http://www.sensorplanet.org/wsw2006/6_Burke_wsw06_ucla_final.pdf

[2] N. D. Lane, S. B. Eisenman, M. Musolesi, E. Miluzzo, and A. T. Campbell, "Urban sensing systems: opportunistic or participatory?" in *Proceedings of the Workshop on Mobile Computing Systems and Applications (HotMobile)*. ACM Press, Feb. 2008, pp. 11–16. DOI: 10.1145/1411759.1411763

[3] S. B. Eisenman, "People-centric mobile sensing networks," Ph.D. dissertation, Columbia University, 2008. Available online: http://comet.columbia.edu/armstrong/pubs/shane-thesis.pdf

[4] S. B. Eisenman, N. D. Lane, and A. T. Campbell, "Techniques for improving opportunistic sensor networking performance," in *Proceedings of the International Conference on Distributed Computing in Sensor Networks (DCOSS)*, Jun. 2008, pp. 157–175. DOI: 10.1007/978-3-540-69170-9_11

[5] S. B. Eisenman, E. Miluzzo, N. D. Lane, R. A. Peterson, G.-S. Ahn, and A. T. Campbell, "The BikeNet mobile sensing system for cyclist experience mapping," in *Proceedings of the International Conference on Embedded Networked Sensor Systems (SenSys)*. ACM Press, Nov. 2007, pp. 87–101. DOI: 10.1145/1322263.1322273

[6] S. Kang, J. Lee, H. Jang, H. Lee, Y. Lee, S. Park, T. Park, and J. Song, "SeeMon: scalable and energy-efficient context monitoring framework for sensor-rich mobile environments," in *Proceedings of the International Conference on Mobile Systems, Applications and Services (MobiSys)*. ACM Press, Jun. 2008, pp. 267–280. DOI: 10.1145/1378600.1378630

[7] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: a scalable and robust communication paradigm for sensor networks," in *Proceedings of the ACM International Conference on Mobile Computing and Networking (MobiCom)*. ACM Press, Aug. 2000, pp. 56–67. DOI: 10.1145/345910.345920

[8] F. Bouhafs, M. Merabti, and H. Mokhtar, "Mobile event monitoring protocol for wireless sensor networks," in *Proceedings of the International Conference on Advanced Information Networking and Applications Workshops (AINAW)*. IEEE Press, May 2007, pp. 864–869. DOI: 10.1109/AINAW.2007.254

[9] G. Werner-allen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh, "Fidelity and yield in a volcano monitoring sensor network," in *Proceedings of the Symposium on Operating Systems Design and Implementation (OSDI)*. USENIX, Nov. 2006. Available online: http://www.usenix.org/events/osdi06/tech/full_papers/werner-allen/werner-allen.pdf

[10] Y. Yao and J. Gehrke, "The Cougar approach to in-network query processing in sensor networks," *SIGMOD Record*, vol. 31, no. 3, pp. 9–18, Sep. 2002. DOI: 10.1145/601858.601861

[11] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TAG: A Tiny AGgregation service for ad-hoc sensor networks," in *Proceedings of the Symposium on Operating Systems Design and Implementation (OSDI)*. ACM Press, Aug. 2002, pp. 131–146. DOI: 10.1145/844128.844142

[12] D. J. Abadi, S. Madden, and W. Lindner, "REED: Robust, efficient filtering and event detection in sensor networks," in *Proceedings of the International Conference on Very Large Data Bases (VLDB)*. VLDB Endowment, Aug. 2005, pp. 769–780. Available online: http://portal.acm.org/citation.cfm?id=1083592.1083681

[13] M. Li, Y. Liu, and L. Chen, "Non-threshold based event detection for 3D environment monitoring in sensor networks," in *Proceedings of the IEEE International Conference on Distributed Computing Systems (ICDCS)*. IEEE Press, Jul. 2007, pp. 9–16. DOI: 10.1109/ICDCS.2007.123

[14] M. Zoumboulakis and G. Roussos, "Escalation: Complex event detection in wireless sensor networks," in *European Conference on Smart Sensing & Context (EuroSSC)*, ser. Lecture Notes in Computer Science, G. Kortuem, J. Finney, R. Lea, and V. Sundramoorthy, Eds., vol. 4793. Springer-Verlag, Oct. 2007, pp. 270–285. DOI: 10.1007/978-3-540-75696-5_17

[15] A. Silberstein, R. Braynard, and J. Yang, "Constraint chaining: on energy-efficient continuous monitoring in sensor networks," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*. ACM Press, Jun. 2006, pp. 157–168. DOI: 10.1145/1142473.1142492

[16] X. Meng, L. Li, T. Nandagopal, and S. Lu, "Event contour: An efficient and robust mechanism for tasks in sensor networks," UCLA, Tech. Rep. TR-040018, 2004. Available online: http://fmdb.cs.ucla.edu/Treports/040018.pdf

[17] I. Solis and K. Obraczka, "Efficient continuous mapping in sensor networks using isolines," in *Proceedings of the International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous)*. IEEE Press, Jul. 2005, pp. 325–332. DOI: 10.1109/MOBIQUITOUS.2005.26

[18] D. Chu, A. Deshpande, J. M. Hellerstein, and W. Hong, "Approximate data collection in sensor networks using probabilistic models," in *Proceedings of the International Conference on Data Engineering (ICDE)*. IEEE Press, Apr. 2006, pp. 48–59. DOI: 10.1109/ICDE.2006.21

[19] A. T. Campbell, S. B. Eisenman, N. D. Lane, E. Miluzzo, and R. A. Peterson, "People-centric urban sensing," in *Proceedings of the Second Annual International Wireless Internet Conference (WICON)*. ACM Press, Aug. 2006, pp. 18–31. DOI: 10.1145/1234161.1234179

[20] C. Cornelius, A. Kapadia, D. Kotz, D. Peebles, M. Shin, and N. Triandopoulos, "AnonySense: Privacy-aware people-centric sensing," in *Proceedings of the International Conference on Mobile Systems, Applications and Services (MobiSys)*. ACM Press, Jun. 2008, pp. 211–224. DOI: 10.1145/1378600.1378624

[21] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations*, R. E. Miller and J. W. Thatcher, Eds. Plenum Press, 1972, pp. 85–103. Available online: http://www.cs.berkeley.edu/~luca/cs172/karp.pdf

[22] R. Hassin and A. Levin, "A better-than-greedy approximation algorithm for the minimum set cover problem," *SIAM Journal Computing*, vol. 35, no. 1, pp. 189–200, 2005. DOI: 10.1137/S0097539704444750

[23] S. M. Ross, *Introduction to Probability Models, Eighth Edition*. Academic Press, Jan. 2003. Available online: http://www.amazon.ca/gp/product/0125980558

[24] B. W. Silverman, *Density Estimation for Statistics and Data Analysis*. Chapman & Hall/CRC, Apr. 1986. Available online: http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20\&path=ASIN/0412246201

[25] C. Heinz and B. Seeger, "Exploring data streams with nonparametric estimators," in *Proceedings of the 18th International Conference on Scientific and Statistical Database Management (SSDBM)*. IEEE Computer Society, 2006, pp. 261–264. DOI: 10.1109/SSDBM.2006.25

[26] G. Losey, D. Ross, and J. Higa, "Estimating behavioral transition rates: Problems and solutions," *Ethology*, vol. 107, no. 2, pp. 89–110, Dec. 2001. DOI: 10.1046/j.1439-0310.2001.00634.x

[27] B. Brewington, "Observation of changing information sources," Ph.D. dissertation, Thayer School of Engineering, Dartmouth College, Jun. 2000. Available online: http://actcomm.dartmouth.edu/papers/brewington:thesis.ps.gz

[28] S. Madden, "Intel lab data," Web page, Intel, 2004, viewed in January 2009. Available online: http://berkeley.intel-research.net/labdata

[29] D. Cuff, M. Hansen, and J. Kang, "Urban sensing: out of the woods," *Communications of the ACM*, vol. 51, no. 3, pp. 24–33, Mar. 2008. DOI: 10.1145/1325555.1325562