DREAM: On the Reaction Delay in Large Scale Wireless Networks with Mobile Sensors

ShaoJie Tang*, Xiang-Yang Li*§, Jing Yuan†, Cheng Wang‡, GuiHai Chen† and Changjun Jiang‡
*Department of Computer Science, Illinois Institute of Technology, Chicago, IL, 60616
†State Key Laboratory of Novel Software Technology, NanJing University, NanJing, China
‡Department of Computer Science, Tongji University, China
§Institute of Computer Application Technology, Hangzhou Dianzi University, China

***Abstract*—In this work, we present a monitor and rescue system utilizing *hybrid networks* which is a integration of stationary sensor networks and mobile sensor networks: *stationary sensor networks* comprised of large numbers of small, simple, and inexpensive wireless sensors, and the mobile sensor network contains a set of mobile sensors (robots). The static sensors in our network have "monitoring" ability, *i.e.*, any activated static sensor can detect the event as long as its sensing range intersects the event region. And the mobile sensors have "moving" and "rescuing" ability, *e.g.*, they can move toward the event region with limited speed and further perform certain rescuing/processing operations on the event. We can consider the event as a hazard, *e.g.*, wild fire, and the mobile sensors as fireman robots. As soon as the fire is detected by the static sensors, the fireman robots are expected to move from its initial location to the hazard region within minimum latency. We define the reaction delay of the system as the delay from the occurrence of event till at least one mobile sensor reaches the event. In order to satisfy certain reaction delay requirement while minimizing the total cost, we propose a number of deployment strategies for the stationary sensor network and mobile sensor network respectively. We further design a random wake-up scheduling for the static sensors for the sake of energy efficiency. Finally, we propose a pure distributed motion strategy for mobile sensors without reliance on localization services such as GPS, focusing on simple algorithms for distributed decision making and information propagation. We demonstrate the efficacy of our system in simulation, providing empirical results.**

Keywords-**Sensor networks, mobility, detection, delay.**

I. Introduction

The recent advances in both the capabilities and miniaturization of wireless sensors have led to applications such as environmental monitoring [12] [13], event detection [15] and security surveillance [11]. These advances herald the development of systems that can gather and harness information in ways previously unexplored. Sensor networks provide a new way of examining environments of interest, delivering numerous small snapshots over time. While traditional sensor networks cannot physically reconfigure themselves to effect more efficient area coverage, in-depth examination of targets, or dynamic protection against inclement environmental developments. By introducing intelligent, mobile sensors into the stationary sensor networks, it provides us the means to enable sensor networks take an active role in manipulating and interacting with their environment. In this paper, we address the reaction delay problem arisen in the hybrid network. In particular, we assume the hybrid network is composed of static sensor network and mobile sensor network: 1) the *stationary sensor network* contains large numbers of small, simple, and inexpensive wireless sensors, and 2) the *mobile sensor network* contains a set of mobile sensors (robots). The static sensor and mobile sensor can communicate with each other if and only if they are within each other's communication range. For simplicity of analysis, we assume the communication ranges of the static sensor and mobile sensor are same, both are equal to some constant r. The static sensors in our network have "monitoring" ability, *i.e.*, any activated static sensor can detect the event as long as its sensing range intersects the event region. The mobile sensors have "moving" and "rescuing" ability, *e.g.*, they can move toward the event region with limited speed and further perform certain rescuing/processing operations at the destination. For example, we may consider the event as a hazard, *e.g.*, wild fire, and the mobile sensors as fireman robots. As soon as the fire is detected by the static sensors, the mobile sensors (fireman robots) are expected to move from its initial location to the hazard region within minimum delay. We define the reaction delay of the system as the latency from the time when the event happens till at least one mobile sensor has reached the event region. A fundamental challenge is to minimize the total cost, *e.g.*, the energy consumption and the number of deployed mobile sensors, while ensuring the required reaction delay.

II. Background and Assumptions

We make the following assumptions on the system settings, which can be found true in many realistic applications:

Monitoring field: We assume that the monitored field Ω is a square area with the edge length of a meters.

Static Sensors and event region: There are n static wireless sensor nodes $\{V|v_1, v_2, \cdots, v_n\}$ are deployed in the monitoring field randomly at uniform. The network consisting of static sensors is called *stationary sensor network*, we further require that the stationary sensor network should satisfy certain connectivity and coverage requirement. The transmission range of each static sensor is same which is equal to r ($r \ll a$). And the sensing radius of each static sensor is also r. An event $\mathcal{E}$ can be considered as a hazard, *e.g.*,wild fire or gas leaking [14] [13]. And we name the continues region occupied by event $\mathcal{E}$ as *actual event region*. And any activated static sensor can detect the event as long as its sensing range intersects the event region. Note that since every static sensor has positive sensing range. We further define the *valid event region* for event $\mathcal{E}$ as the region within which any activated static sensor can detect $\mathcal{E}$. Obviously, the area of valid event region is always larger than the one of actual event region. To simplify notions, we always use *event region* to represent *valid event region* throughout the paper. We further assume that the arrival distribution of events follows "one by one" manner, that is, at most one event may happen within a sufficient long time duration.

Mobile Sensors: There are m mobile nodes $\{U|u_1, u_2, \cdots, u_m\}$ which can move in the field to perform certain rescuing operation on the event. Mobile sensors initially deployed in a stationary manner and are directed to move toward a possible target only when an event is detected. Such a two-phase strategy allows mobile sensors to avoid unnecessary movement through the coordination with static sensors. The network consisting of mobile nodes is called *mobile sensor network*. The transmission range of each mobile nodes is same as the one of static sensor which is equal to r. However, it will not affect the generality of our results as long as it is no smaller than r. The moving speed of the mobile sensors is equal to s ($s \ll a$) meters per second. The radio transmission speed is much faster than the moving speed of mobile nodes. We assume that a mobile sensor knows the direction (not necessarily location) of a static sensor node if they can directly communicate with each other.

III. Problem Formulation and Approaches

We first give a formal definition for the reaction delay as follows

Definition 1: Suppose an event happens at time slot t_1, it has been detected by the stationary sensor network at time slot t_2, and finally at least one mobile sensor can reach the event region at time slot t_3. Then the detection delay is defined as $T_{detect} = t_2 - t_1$ and the motion delay as $T_{motion} = t_3 - t_2$ and the reaction delay is defined as

$$T = t_3 - t_1 = T_{detect} + T_{motion}$$

As the radio transmission speed is much faster than the movement speed of mobile nodes, for simplicity of analysis, the time spent on the information propagation part is omitted here. Again, it will not affect the generality of our results. The objective of this paper is to find a efficient way to minimize the overall reaction delay. In particular, assume we are given the cost of deploying one mobile sensor, the cost of waking up one static sensor at each time slot and a overall reaction delay requirement. We try to disclose the relationship between the delay bound and different parameters in network designing, *e.g.* number of deployed mobile sensors, the number of activated static sensors at each time slot. Since in our problem setting, the detection delay only depends on the structure of stationary sensor network, thus in order to minimize the detection delay, we focus on the designing for stationary sensor network in Section VI; To address the motion delay, we concentrate on studying the mobile sensor network in Section VII-B. In particular, based on the pre-deployed stationary sensor network, we propose a number of deployment strategies as well as motion strategy for the mobile sensors.

IV. Related Work

Several recent studies [4] [3] analyzed the impact of mobility on detection delay and area coverage. These studies are based on random mobility model and do not address the issue of actively controlling the movement of sensors. Bisnik *et al.* [6] analyzed the performance of detecting stochastic events using mobile sensors. Chin *et al.* [4] proposed to improve the coverage of a region by patrolling static routes using mobile sensors. Xing *et al.* [2] explores the use of mobile sensors to address the limitation of static WSNs for target detection. They develop an optimal sensor movement scheduling algorithm that enables mobile sensors to gather the maximum amount of target energy under a given moving distance bound. S. Gandhi *et al.* [10] propose a scalably efficient scheme for detecting large-scale physically-correlated events in sensor networks. They show that in a network of n sensors arbitrarily distributed in the plane, a sample of $O(\frac{1}{\epsilon}\log\frac{1}{\epsilon})$ sensor nodes is sufficient to catch any, and only those, events that affect $\Omega(\epsilon n)$ nodes, for any $0 < \epsilon < 1$, as long as the geometry of the event has a bounded Vapnik-Chervonenkis (VC) dimension.

V. Technical Lemmas

Lemma 1 (Binomial Distribution): Consider n independent variables $X_i \in \{0,1\}$, $p = \mathbf{Pr}(X_i = 1)$, and $X = \sum_{i=1}^{n} X_i$.

$$\Pr(X > \xi) < \frac{\xi(1-p)}{(\xi - n\cdot p)^2}, \qquad \text{when } \xi > n\cdot p.$$
$$\Pr(X \le \xi) \le \frac{(n-\xi)p}{(np-\xi)^2}, \qquad \text{when } 0 < \xi \le n\cdot p.$$

We recall the following definitions by Vapnik and Chervonenkis [16]. Let $\mathcal{U}$ be the input space. Let $\mathcal{C}$ be a family of subsets of $\mathcal{U}$. A *finite* set S (called sample in machine learning) is *shattered* by $\mathcal{C}$, if for *every* subset B of S, there *exists* a set $A \in \mathcal{C}$ such that $A \bigcap S = B$. The *VC-dimension* of $\mathcal{C}$, denoted by VC-d($\mathcal{C}$), is defined as the maximum value d such that there exists a set S with cardinality d that can be shattered by $\mathcal{C}$. For sets of *finite* VC-dimension, one has uniform convergence ini the weak law of large numbers:

Theorem 2 (The Vapnik-Chervonenkis Theorem): If $\mathcal{C}$ is a set of *finite* VC-dimension VC-d($\mathcal{C}$), and $\{X_i \mid i = 1, 2\cdots, N\}$ is a sequence of *i.i.d.* random variables with *common* probability distribution P, then for every $\epsilon, \delta > 0$, $\mathbf{Pr}\left(\sup_{A\in C}\left|\frac{\sum_{i=1}^{N} I(X_i \in A)}{N} - P(A)\right| \le \epsilon\right) > 1-\delta$ whenever $N > \max\left\{\frac{8\cdot \text{VC-d}(\mathcal{C})}{\epsilon}\cdot\log\frac{13}{\epsilon}, \frac{4}{\epsilon}\log\frac{2}{\delta}\right\}$. Here $I(X_i \in A)$ takes value 1 if $X_i \in A$ and 0 otherwise.

Lemma 3: If we randomly place $(1+\varepsilon)n\ln n$ balls into n bins, then all the n bins are occupied with probability at least $1 - \frac{1}{n^{\varepsilon}}$, where $0 < \varepsilon < 1$ is a constant.

NOTATIONS: Throughput this paper, for a continuous region Ω, we use $|\Omega|$ to denote its area; for a discrete set S, we use

$|S|$ to denote its cardinality; for a tree T, we use $\|T\|$ to denote its total Euclidean edge lengths; $x \to \infty$ denotes that variable x takes value to infinity.

VI. Stationary Sensor Network

In order to satisfy certain detection delay requirement, we propose a deployment strategy and random wake up scheduling for static sensors in this section.

A. Deployment Strategy

Recall that in this paper, we consider a square monitoring region of size $a \times a$. We assume all static sensors have the same communication range r. One natural question is that how many nodes are needed to construct a connected network if all nodes are placed in the region randomly and uniformly. Gupta and Kumar [17] studied the critical transmission range (CTR) for connectivity: what is the CTR for connectivity if we randomly deploy n sensor nodes in a unit square. Their pioneering work showed that if n nodes V are uniformly randomly deployed in a unit square and the transmission range r satisfies that $n\pi r^2 \geq c_1 \ln n$ for some constant c_1, then the network $G(V, r)$ (two nodes of V are connected in G if and only if their Euclidean distance is at most r) will be connected with probability 1 when $n \to \infty$. In addition, if $n\pi r^2 \leq c_0 \ln n$ for some special constant c_0, the network $G(V, r)$ will be disconnected with a positive constant probability for $n \to \infty$. This implies that

When all static sensors have transmission range r, the asymptotic number n of nodes needed to deploy in a square of size $a \times a$ such that the resultant network is connected satisfies that $n\pi(\frac{r}{a})^2 = \Theta(\ln n)$. Let $h = (\frac{a}{r})^2$. Therefore $n = \Theta(h \ln h)$. Another similar problem is to study the number of random nodes to deploy in the square region such that the entire region is covered by the disks centered at the sensors with radius r (called **sensing disk** for each node). Previous results proved that it costs $n = \Theta(h \ln h)$ nodes to cover the target square area with high probability when all sensor nodes are deployed randomly at uniform. Consequently, we have

Theorem 4: [17] If we randomly deploy $n = \Theta(h \ln h)$ static sensors in the square region with side length a, where $h = (\frac{a}{r})^2$, the resultant network will be connected and the square region will be covered by the sensors with high probability.

Next, we will prove that after we deploy n static sensors, any event region with area $\Omega(\mathcal{E})$ will contain sufficient large number of static sensors with high probability. Given a event region $\Omega(\mathcal{E})$, we use $|\Omega(\mathcal{E})|$ to denote its area and $V(\mathcal{E})$ to denote the set of sensors deployed inside $\Omega(\mathcal{E})$.

Theorem 5: After we deploy n static sensors in the monitoring field Ω randomly at uniform, for any event region with area at least $|\Omega(\mathcal{E})| \geq \alpha a^2$, we have

$$\Pr(|V(\mathcal{E})| > c_1\alpha \cdot n) \geq 1 - \theta_1 \cdot \frac{1}{n}$$

where $0 < c_1 \leq 1$ is some constants and $\theta_1 = \frac{1 - c_1\alpha}{(1-c_1)^2\alpha}$.

Proof: According to the property of Lemma 1, that is, X is a binomially distributed random variable, n is number of trials,p is success probability. In our specific case, $X = |V(\mathcal{E})|$ is number of static sensors fallen in the event region $\Omega(\mathcal{E})$, n is total number of static sensors randomly distributed in the monitoring region Ω, p is the probability of a node falling in the event region $\Omega(\mathcal{E})$. Note that $p = \frac{|\Omega_i|}{a^2} = \alpha$, it follows that,

$$\Pr(|V(\mathcal{E})| \leq c_1\alpha \cdot n) \leq \frac{(n - c_1\alpha \cdot n)\alpha}{(\alpha \cdot n - c_1\alpha \cdot n)^2}, \quad \text{when } 0 < c_1 \leq 1.$$

This finishes the proof. ■

As shown in Theorem 5, for any event region $\Omega(\mathcal{E})$ with area at least αa^2, there exists almost surely at least $c_1\alpha \cdot n$ sensors that falling inside it.

B. Adaptive Random Wake-Up Scheduling

Based on the previous deployment strategy, its obvious that if *all* the static sensors are activated *all* the time, any event with sufficient large area can be detected immediately, *e.g.*, $T_{detect} = 0$, with high probability. However, in some scenarios, we may allow certain detection delay, *e.g.* $T_{detect} \leq t_d$. Accordingly, instead of letting all the static sensors wake up all the time, we may only need to let partial of the static sensors active per time slot for the sake of energy efficiency. Here we name the set of activated static sensors as sampling set. In the following contents, we derive a general function to disclose the relationship between the detection delay bound and the size of sampling set per time slot.

Theorem 6: When the area of event region $\Omega(\mathcal{E})$ is at least $|\Omega(\mathcal{E})| \geq \alpha a^2$, if we randomly pick k static sensors to wake up at each time slot, we have

$$\Pr(T_{detect} \leq t_d) \geq \left(1 - c_2^{kt_d}\right)(1 - \frac{\theta_1}{n})$$

where $c_2 < 1$ and θ_1 are some constant.

Proof: Proof is omitted here to save space. ■

C. Discussion on the false alarming

In our previous analysis, we assume that the system decision will be made as long as one active static sensor detect the event. This scheme suffers from the false alarming, that is the static sensor making a positive detection decision when no event is present [19]. In our designed system, the final system decision is made by the static sensor which detects the event initially as soon as enough local decisions are received to reach a majority consensus. Specially, if some static sensor v detects an event, it will wake up all its immediate neighbor sensors (one-hop away neighbor in the communication graph). Among all its neighbor sensors, any sensor who else detects the event also wakes up its neighbor sensors. As this process going on, as long as the total number of static sensors which detect the event is larger than threshold value γ, the event is confirmed and all the sensors which detects the event will become POI (point of interest). Here γ is chosen accordingly in order to reduce the false alarming. These POI sensor nodes will further raise and spread the alarm through the network, this part will be discussed in details in the following sections.

VII. Mobile Sensor Network

In this section, we propose a number of deployment and motion strategies for the mobile sensors in order to meet certain motion delay requirement.

A. Deployment Strategy

For simplicity of analysis, we assume that there exists a perfect navigation algorithm that can guide the mobile sensor moving *directly* from one static sensor to another one, *e.g.*, along a straight line. Further, we are given a motion delay bound t_m such that $T_{motion} \leq t_m$. For example, if the event is detected at time slot t_2, it is required that at least one mobile sensor is able to be guided to that static sensor no later than $t_2 + t_m$. In this section, we mainly address the following problems: How many mobile sensors are needed to satisfy certain motion delay requirement?

Given the the pre-deployed static sensor network and motion delay bound t_m, how to find a feasible deployment for the mobile sensors such that the number of required mobile sensors is minimized? Here a feasible deployment strategy is the one under which the given motion delay requirement is satisfied. As shown in Lemma **??**, under any feasible deployment strategy, there must exist at least one mobile sensor within $\mathcal{D}(v_j, st_m)$ for any v_j.

Clearly, the mobile sensors can be placed at any point in the two-dimensional plane. Without any constraint, it will lead to a infinite input size. In order to bound the time complexity, we would expect that the possible deployment locations for the mobile sensors can be narrowed down to a set of points with limited size. Fortunately, our next theorem show that by restricting the possible deployment positions for mobile sensors to the locations of pre-deployed static sensors, we can still get a constant approximation solution. In other words, instead of considering all possible points in the 2D monitoring field, we only need to study the points which are already occupied by the pre-deployed static sensors. This is formally stated in the following lemma.

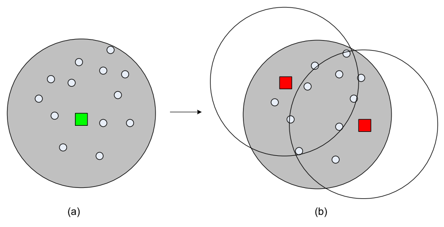

Fig. 1. By replacing any mobile sensor (green square) at arbitrary position using at most 9 mobile sensors (red squares) positioned together with some static sensors, all the static sensors can still be supported.

Lemma 7: There always exists a feasible deployment strategy in which all the mobile sensors are placed together with some static sensors such that the total number of required mobile sensors is at most 9 times the optimum one.

Proof: We prove it through construction. Given a optimum deployment strategy, we call the set of static sensors fallen in disk $\mathcal{D}(u_i, st_m)$ as the *support set* of u_i for any mobile sensor u_i. The support set of u_i simply means the set of static sensors that can be reached by u_i within time t_m under perfect navigation scheme. Clearly, every static sensor must belong to at least one mobile sensors' support set in any feasible solution. Assume that there exists some mobile sensor u_i in the optimum solution where it is not deployed with some static sensors but some arbitrary point in the monitoring field. See Fig 1 (a) for example. If we remove u_i from the optimum solution, there must exist some static sensors which is not supported by any other mobile sensor. Then we randomly choose one of those unsupported static sensors to place a mobile sensor on it. We keep adding new mobile sensors to the remaining unsupported static sensors until every static sensor can be supported by some new added mobile sensors. By simple area argument, we know that there exists at most $\pi(st_m + \frac{st_m}{2})^2/\pi(\frac{st_m}{2}^2) = 9$ disjoint disks with radius $st_m/2$ contained in disk $\mathcal{D}(u_i, st_m + \frac{st_m}{2})$. It implies that in order to replace any mobile sensor in the optimum solution, it is sufficient to place 9 additional mobile sensors together with some static sensors such that all the static sensors can still be supported. This finishes the proof. ■

We next concentrate on finding a near optimum solution under the location constrains, that is, the mobile sensors are only allowed to place together with some static sensors. Lemma 7 shows that that the number of required mobile sensors in our near optimum solution is at most 9 times the minimum one.

First, based on the pre-deployed static sensor network, we construct a new graph $G = (V, E)$ where $V = \{v_1, \cdots, v_n\}$ is the set of static sensors. For any pair of vertices v_i, $v_j \in V$, if $||v_i - v_j|| \leq st_m$, we add an edge between them in G. Then our problem become equivalent to finding a minimum dominating set(MDS) in the new constructed graph. Unfortunately, the original MDS problem is *NP*-Hard, even on unit disk graphs(UDG) where a geometric representation is given [7].

Most of the work concerning approximation schemes in UDGs assume a given geometric representation, which allows for separation of the graph along a grid [9]. However, in order to be well suited for the implementation in wireless network, we find a approach which requires no geometric coordinates. In [8], local neighborhoods of limited graph-theoretic diameter are used to obtain a PTAS for the maximum independent set problem in the same setting. This method uses the fact that in such neighborhoods, a maximum independent set is of bounded cardinality. In this paper, the same fact is used to bound the size of a minimum dominating set. And we further prove that our algorithm has polynomial time complexity. The basic idea of the construction is simple: By random selecting a vertex $v \in V$, we compute a local dominating set for the neighborhood of v, and expand this neighborhood until we have formed sets S and $T \supset S$ which satisfy a desired bound. Then, we eliminate the current neighborhood and continue the same steps for the remaining graph. In more detail, the algorithm works as follows. We start with an arbitrary vertex $v \in V$ and consider for $g = 0, 1, 2, \cdots$,

Algorithm 1 Near Optimum Deterministic Deployment

Input: Monitoring field Ω, a growth-bounded graphs $G = (V, E)$ resulted from the pre-deployed static sensor network.
Output: Subset of V which serve as the initial position for mobile sensors.

```
1:  $V' := V$
2:  **while** $V' \neq \emptyset$: **do**
3:      Randomly pick a vertex $v$ from $V'$;
4:      $g = 0$;
5:      **while** $|D(N^{g+2}(v))| > \rho|D(N^{g}(v))|$ **do**
6:          $g := g + 1$;
7:      **end while**
8:      We compute a local dominating set $D(N^{g+2}(v))$ for
        the neighborhood of the vertex $v$;
9:      $V' := V'/D(N^{g+2}(v))$;
10: **end while**
11: $D(V)$= Union all the local dominating set computed
    above;
12: The mobile sensors will be deployed at the same position
    as the static sensors in $D(V)$.
```

the g-th neighborhoods $N^{g}(v)$. Starting with $N^{0}(v) = v$, we compute dominating sets of minimum cardinality for these neighborhoods as long as $|D(N^{g+2}(v))| > \rho|D(N^{g}(v))|$ holds.

Theorem 8: The above algorithm returns a deployment strategy for mobile sensors such that the total number of required mobile sensors is at most 9ρ times the optimum solution.

Sketch of proof: Basically, we can prove that by following our algorithm, the number of deployed mobile sensors is at most ρ times the one under the near optimum solution (where the deployment locations are constrained). And together with the fact that the number of mobile sensors required in the near optimum solution is at most 9 times the real optimum one. Our theorem follows.

In order to evaluate the time complexity of our algorithm, we first give the following lemma:

Lemma 9: Any independent set $I^{g} \subset N^{g}(v), v \in V$, satisfies $|I^{g}| \leq (2g+1)^{2}$.

Proof: I^{g} consists of pairwise disjoint disks of radius at least $st_m/2$ inside a disk of radius at most $g \cdot st_m + st_m/2$, and therefore $|I^{g}| \leq \pi(g+\frac{1}{2})^{2}/\pi(\frac{1}{2})^{2} = (2g+1)^{2}$. ∎

Since the maximal independent set is a special dominating set, therefore, the size of minimum dominating set must be upper bounded by the size of maximal independent set. It follows that

Lemma 10: There exists a constant $\theta = \theta(\rho)$ such that $g \leq \theta$, that is, the largest neighborhood to be considered during the iteration of the algorithm is bounded by a constant.

Proof: Proof is omitted here due to limited space. ∎

B. Motion Strategy For Mobile Sensors

In this section, we propose a distributed motion strategy for mobile sensors. Our basic idea is to build a virtual 3D map in the stationary sensor network, with POIs locating at the "valley bottom". Mobile nodes thus fall to the "valley bottom" with the help of local information, *e.g.*, the height of the neighboring sensors in the virtual 3D map. The information at a static sensor that is used to guide the movement of mobile nodes is called the potential value of the sensor, which is denoted as $\mathtt{p}(v_i)$ for each static sensor v_i.

In this paper, we adopt a simple Breath First Search (BFS) Tree based method to calculate the potential value of each static sensor. In particular, when a static sensor v confirms a event, it becomes a POI. Based on the communication graph, we try to set the potential value for each static sensor v_i as its hop distance from v. Note that, there may be more than one static sensors detect the event and becomes POIs at the same time. And the potential value of static sensor v_i is determined by the hop distance from its closest POI. Further, the v_i will be guided to its closest POI.

Initially, all the static sensors $v_i \in V$ set their potentials to $\mathtt{p}(v_i) = n$. Without detecting any event, the initial potential field is flat. When a static sensor v_d detects a event, it first confirms it with its nearby sensors as explained before. Immediate after the confirmation, it arises the alarm and spread it through the whole network. In order to be well suited for the piratical implementation, especially when there is no central control, we build the potential field in a pure distributed way as shown in Algorithm 2.

It is easy to find the potential value of each static sensor becomes its hop distance from its closest POI. Clearly, the POI will have the minimum potential among all its neighbors and a non-POI sensor closer to a POI obtains a lower potential than other non-POI sensors far away from the POI. Guided with the potential values of the nearby static sensors, a mobile node moving from a sensor of high potential toward a sensor of lower potential and will finally reach the target.

Lemma 11: By following our motion strategy, the mobile sensor u_i can reach the event region through a path with minimum hop distance in the communication graph.

Proof: Since the potential value of each static sensor is the hop distance from its closest POI. Assume that among all the static sensors in the communication range of u_i, v_i has the lowest potential value. Then it must have the minimum hop distance from the event region. Since during each step, u_i will move from one static sensor to another one with lower potential value, the total number of moving steps is at most $\mathtt{p}(v_i)$. Therefore by following our motion strategy, the mobile sensor can always reach the event region along a path with minimum hop distance. ∎

In this way, without any central control or global information, our approach guarantees that a mobile node moves toward a correct direction, that is, a direction including sensors that have the lowest potential value 0. A mobile node will not be stuck at a sensor unless this sensor is a v_d.

C. Discussion On the Motion Delay

In this section, we address the exact motion delay resulted from our deployment and motion strategy. Remember that in

Algorithm 2 Distributed Motion Strategy For Mobile Sensors

```
1:  Potential Field Built-Up Phase for each $v_i$:
2:  $\mathtt{p}(v_i) = n$;
3:  if $v_i$ detects and confirms a event then
4:      $\mathtt{p}(v_i) = 0$;
5:      Sending updating message containing $\mathtt{p}(v_i)$ to all its
        direct neighbors $N(v_i)$;
6:  end if
7:  if $v_i$ receives a updating message from $v_j$ then
8:      if $\mathtt{p}(v_i) > \mathtt{p}(v_j) + 1$ then
9:          $\mathtt{p}(v_i) = \mathtt{p}(v_j) + 1$;
10:     end if
11: end if
12: Motion Strategy for each $u_i$:
13: $u_i$ will periodically check the potential value of each static
    sensor in its communication range;
14: if $u_i$ finds a static sensor $v_j$ which has the lowest potential
    value then
15:     it will move toward $v_j$ until it reaches it;
16: end if
```

Section VII-A, we represent our deployment strategy based on following assumptions: (1) There exists a perfect navigation algorithm that can guides a mobile node to move directly to the POI; (2) the potential field can be built up through information propagation in the network. Based on the fact that our pre-deployed stationary sensor network is connected, the second assumption is easily achieved. Since the speed of radio transmission is much faster than the moving speed of mobile sensors, for simplicity of analysis, the delay happens in the information propagation phase is omitted here. But it will affect the generality of our results. Unfortunately, our motion strategy is obviously not a "perfect" one. In other words, by following Algorithm 2, the mobile sensor can hardly move *directly* to the static sensor(except the static sensor within communication range) along a straight line. Instead, the moving path of a mobile sensor is more like a continuous line composed of one or more line segments. See Fig VII-C for illustration. This is because in our navigation scheme, in order to access and update the movement instructions and further follow the directions, the mobile sensors should get sufficient close to the static sensors along its moving path.

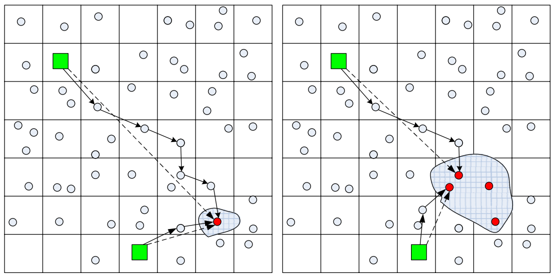
Fig. 2. The moving path by following our navigation scheme (bold line) and perfect scheme (dot line).

We next show that the length of the actual moving path from the initial position to the destination is no larger than $\sqrt{2}$ times the euclidian distance between them with high probability. To address this issue, we first partition the region into squarelets, each of side-length $\sqrt{2}r/4$. Since we assume the transmission range of the mobile sensor is r, then the mobile sensor can communicate with any static sensor in the 8 adjacent cell directly. Thus, by querying the static sensor from the squarelet, the mobile sensor can be navigated from one squarelet to the any adjacent squarelet (sharing a common point) directly.

Lemma 12: There is a sequence of $\delta(n) \to 0$ such that

$$\mathbf{Pr}\,(\text{Every squarelet contains a static sensor}) \geq 1 - \delta(n)$$

Proof: Let $\mathcal{C}$ be the class of axis-aligned squares of side-length $\sqrt{2}r/4$. Notice that the probability that a static sensor fall in such a square is $\frac{r^2}{8} \cdot \frac{1}{a^2} = \frac{r^2}{8a^2}$. Recall that, to have a connected stationary sensor network, we almost surely have $r/a \geq \Theta(\sqrt{\frac{\log n}{n}})$. It is easy to show that the VC-dimension of $\mathcal{C}$ is at most 4 (it is at least 3). Hence, for all squarelets S, $\mathbf{Pr}\left(\sup_{S\in\mathcal{C}}\left|\frac{\#\text{ of nodes in } S}{n} - \frac{r^2}{8a^2}\right| \leq \epsilon(n)\right) > 1 - \delta(n)$ whenever

$$n \geq \max\left\{\frac{32}{\epsilon(n)} \cdot \log\frac{13}{\epsilon(n)}, \frac{4}{\epsilon(n)}\log\frac{2}{\delta(n)}\right\}. \qquad (1)$$

This condition 1 is satisfied when $\epsilon(n) = \frac{32\log n}{n}, \delta(n) = \frac{2}{n}$. Thus, $\mathbf{Pr}\left(\sup_{S\in\mathcal{C}}\{\#\text{ of nodes in } S \geq \frac{nr^2}{8a^2} - n\cdot\epsilon(n)\}\right) > 1 - \delta(n)$. Thus, if we choose n such that $r \geq 17 \cdot a \cdot \sqrt{\frac{\log n}{n}}$, then $\frac{nr^2}{8a^2} - n\cdot\epsilon(n) \geq 4\log n$. Consequently, we have

$$\mathbf{Pr}\,(\forall \text{ squarelet } S, \#\text{ of nodes in } S \geq 4\log n) > 1 - \frac{2}{n}$$

The theorem then follows. ■

Notice that, generally, when $r = a\sqrt{\frac{\log n}{c\cdot n}}$, to make sure that every squarelet with side-length r/β will have at least one node inside, it is sufficient to require that $c < \frac{1}{32\beta^2}$.

Theorem 13: By following Algorithm 2, the mobile sensor u_i can be guided to its closet POI node v_j along a path within length $\sqrt{2}\cdot||u_i - v_j||$ with high probability.

Proof: In light of Lemma 12, we know that almost surely there exist at least one static sensor in each squarelet. Therefore, with high probability, the minimum hop-distance between u_i and v_j is at most $\sqrt{2}\cdot\frac{||u_i-v_j||}{r}$. As shown in Lemma 11, we prove that by following our motion strategy, the mobile sensor will reach its closet POI along a path with minimum hop distance. It follows that the length of actual path is at most $\sqrt{2}\cdot\frac{||u_i-v_j||}{r}\times r = \sqrt{2}\cdot||u_i - v_j||$ with high probability. ■

Remember that throughout the discussions in Section VII-A, we always assume there exists a perfect navigation algorithm. In light of Theorem 13, we know that by following our specific motion strategy, the moving distance is at most $\sqrt{2}$ times the euclidian distance with high probability. Thus in order to satisfy the motion delay t_m with high probability, it is sufficient to let the delay requirement in the deployment strategy proposed in Section VII-A be $t_m/\sqrt{2}$ instead of t_m.

VIII. Discussion

Discussion on the Overall Delay: Remember that the objective of this paper is to find a efficient way to minimize the overall reaction delay T, that is the sum of detection delay T_{detect} and motion delay T_{motion}. In particular, assume we are given the cost of one mobile sensor, the cost of waking up one static sensor at each time slot and a overall reaction delay requirement. We try to find a trade off between the size of sampling on static sensors at each time slot and the number of deployed mobile sensors. In the previous sections, we disclose the relationship between the size of sampling set and the detection delay; the size of deployed mobile sensors and motion delay respectively. Therefore our previous results are able to serve as the guideline in parameter selection in order to minimize the total cost.

Self-Organization Mobile Sensor Redeployment: Note that all previous discussions concentrate on finding a initial deployment and motion strategy for the mobile sensors under the assumption that only one event need to be rescued. However, after the one event has been processed/rescued, the current deployment of the mobile sensors may differ from the initial one significantly without redeployment, *e.g.*, most of them gathered at the previous event region. In order to maintain the initial coverage, it is necessary to redeploy all the mobile sensors. One of the possible ways is based on the method proposed in [5]. Basically, we first divide the monitoring field Ω into cells. Since after the mobile sensors gathers around the event region, there must exist a large fraction of cells that contains no mobile sensors. The goal of redeployment is to let all the mobile sensors be positioned uniformly in each cell, then we build up a one-to-one matching between mobile sensors and vacancies such that the moving distance is minimized. By implementing the push-relabel algorithm [5], all the vacancies can be filled in a distributed way.

IX. Simulation Results

We conducted extensive simulations to study the performance of our event handling schemes in terms of scalability and total delay. We begin by describing the general methodology and parameters of our simulation study.

A. Methodology and Parameters

We distribute static nodes uniformly at random in a square region with side length a. Random events are generated with α, indicating the size of the area, ranging from 0.01 to 0.05. In our experiments, we set the probability that a target event appears at the beginning of a sampling interval to be 5%. Further, each target appearance is assumed to last for 15 seconds, in other words, we have to detect the event and navigate at least one mobile sensor to the POI within 15 seconds with high probability. Here δ, which indicates the probability to miss an event, is set to be 0.05. The transmitting range of static nodes, r, is set to be 10m as a default value. Recall that we assume the communication range of mobile nodes is same as r. To study the performance of our node deployment strategy, the detection and motion delay are collected with different event size and different moving speed. In our collection of the motion delay, the random and the near optimum deterministic deployment strategies are performed, respectively. After some static sensor detects an event, the mobile sensors will move toward that sensor by following the same motion strategy. According to the typical speed of mobile sensor systems, we set the moving speed on the monitoring field ranging from 0.2m/s to 1m/s.

In our experiments, the sampling sets are constructed by using a redundancy-aware scheme, where any sample that is the same with a preciously chosen sample is discarded, and resample. In this way, we improve the size of the sampling set. We use circlesas the event geometry in our experiments. To generate a circle event, we choose its center uniformly at random in the monitoring region, and then increase the radius until the area of the disk reaches the target size. We generated thousands of random events to evaluate the performance for each of our experiments. Thus, each of our experiments reports the cumulative outcome of these tests. Now we discuss the results of our simulations.

B. Detection Delay

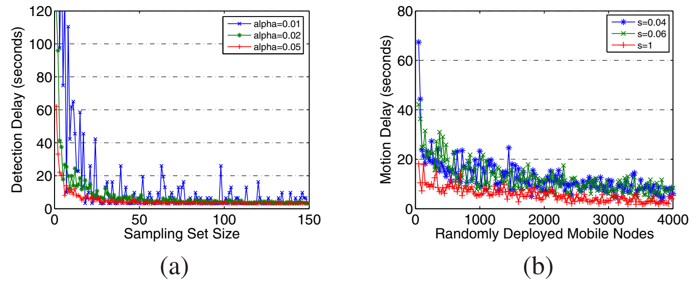

Fig. 3. Detection delay with the increase of the sampling set size q and different α; motion delay with the increasing number of randomly deployed mobile nodes and different mobile speed s.

Our first set of experiments evaluate the basic performance of our random sampling based detection model in terms of the detection delay. The detection delay, T_{detect}, is computed as a time duration between the occurrence of the target event and a detection consensus is reached by all the nearby static sensors. We fix the number of mobile sensors to collect the detection delay with the increasing size of the sampling set, as showed in Figure 3(a). In our experiments, we assume a time duration, t, is needed to conduct the next sampling set and collect the measurement data back to Base Station for each round of samplings. Here t is set to be 3.25 seconds according to empirical values. The detection delay will be slightly lower if we omit this duration, in which case each sensor is assumed to be able to make a local decision of waking up or not for each round of samplings. As expected, the detection delay grows inversely proportional to the sampling set size. Figure 3(a) shows that all three curves converge to below 10 seconds, indicating that we need at most three sampling sets with appropriate sizes (150 samples) to detect the target events in the monitoring region with high probability. Moreover, larger sampling sets are needed for smaller α since it indicates a less significant event. In the following experiments, we deploy mobile sensors based on this sampling-assisted detection model to evaluate the performance of our mobile node deployment schemes and motion strategy.

C. Motion Delay

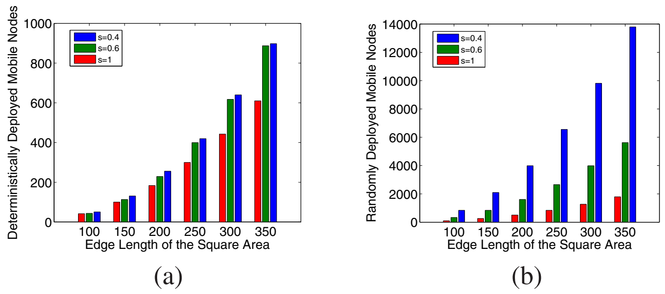

Fig. 4. Number of required mobile sensors under motion delay requirement $t_m = 10$s with the increasing network size a and different moving speed s.

In this set of experiments, we evaluate the performance of our mobile sensor deployment scheme and the effectiveness of our motion scheduling algorithm in terms of the motion delay. The motion delay, T_{motion}, is computed as a time duration between some static sensor detects the event and at least one mobile sensor reaches the nearest POI. We fix the number of sampling sensors to collect the motion delay with the increasing number of mobile sensors deployed.

1) Random Deployment: We vary the number of uniformly deployed mobile sensors from 50 up to 1500, and for each number, 100 events are generated uniformly at random in the region of size 200×200 m^2 to estimate T_{motion}. As Figure 3(b) showed, the motion delay grows inversely proportional to the number of randomly deployed mobile nodes, and more mobile nodes are needed to satisfy the delay requirement as the moving speed is lower. It is clearly showed that at most 600 mobile nodes suffice when the moving speed reaches 1m/s to guide at least one mobile sensor to the nearest POI within 10 seconds.

2) Near Optimum Deterministic Deployment: When near optimum deterministic deployment strategy is employed, we can not adjust the number of mobile sensors to collect the motion delay, since all the mobile nodes are carefully placed at the same location of some static sensors. Here we design another experiment to illustrate the number of mobile nodes needed to satisfy the motion delay for a scaling network size. As Figure 4(a) showed, when the network size increases, more mobile nodes are needed to make sure that at least one mobile node reaches the POI within t_m. It is clearly showed that, with a=200, at most 200 mobile nodes suffice when the moving speed reaches 1m/s to satisfy the delay requirement. In contrast to the random deployment scenario, by following our deterministic deployment scheme, the number of needed mobile nodes is dramatically reduced. More in detail, our deterministic scheme reduces the number of deployed mobile nodes by at least 90%, 80% and 70%, respectively, as moving speed s ranges from 0.04m/s to 1m/s. This result shows that our deterministic deployment algorithm can significantly reduce the manufacturing overhead by taking advantage of carefully choosing deploy positions.

3) Impact of Mobile Sensor Speed: We also evaluated the impact of mobile sensor speed on the system performance. We illustrate the motion delay as the moving speed, s, ranging from 0.4m/s to 1.0m/s. As showed in Figure 4(a)(b), when the speed grows higher, the number of mobile sensors needed to satisfy the delay requirement decreases considerably. This result shows that our motion scheduling algorithm can effectively improve the detection performance by taking advantage of the increase of the moving speed.

D. Total Delay

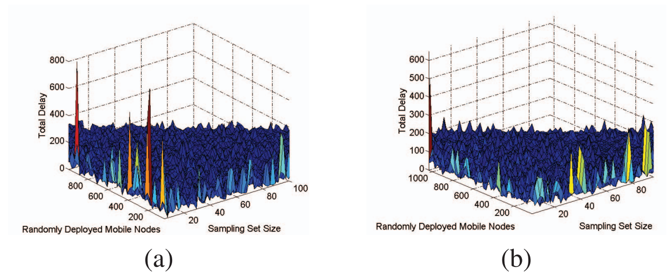

Fig. 5. Total delay with increasing number of randomly deployed mobile nodes and increasing sampling set size ((a)α=0.02, s=0.8m/s);(b) α=0.05, s=0.8m/s).

In this experiment, we illustrate the overall reaction delay with increasing number of randomly deployed mobile nodes and different sampling set size. Different from the previous experiments, we deploy static sensors uniformly at random in a larger square region of size 500×500 m^2, where the number of static nodes reaches 4000. We set the moving speed s as 0.8m/s. The total delay collected with different target event size is illustrated in Figure 5(a) and Figure 5(b). As expected, when the number of static and mobile nodes increases, the total delay decreases considerably from more than 200 seconds to below 50 seconds. Moreover, the overall delay is also decreased by almost 50% as we extend the event scale by adjusting α from 0.02 to 0.05.

E. Examples

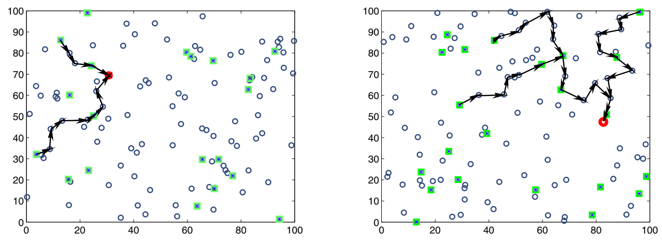

Fig. 6. The moving paths of mobile sensors by following our motion strategy when a single sample hits the target.

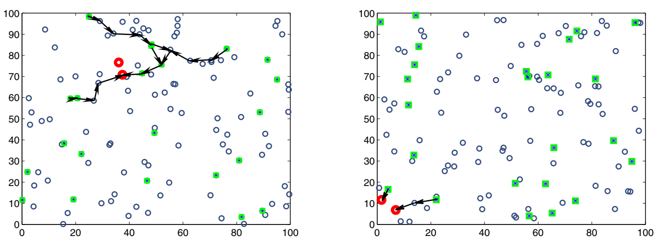

Fig. 7. The moving paths of mobile sensors by following our motion strategy when multi samples hit the target.

Here we conducted more illustrations for evaluation the performance and effectiveness of our sensor deployment schemes and movement scheduling algorithms, as showed in Figure 6 and Figure 7. For simplicity, a monitoring region of size $100 \times 100\ m^2$ is used in this set of experiments. Mobile nodes are deterministically deployed in the field and are marked as blue circles with a green square boundary.As described above, static nodes, as blue circles showed in the figures, are deployed uniformly at random in the square. After some static node(s) hit the target event (marked as red circles), the mobile nodes that detect a lower potential value of neighbors can keep moving toward the POI based on repeatedly checking the lowest potential values of new neighbors. Then, the moving paths are drawn up to show which nodes and how they finally reach the POI. Mobile nodes without a moving path keep still at its original position since the event information did not reach them, which is unnecessary in realistic. In this way, our detect and rescue schemes effectively locate the target event merely based on local information, and only necessary motion of nearby mobile nodes is conducted for the rescue work. In addition, the mobile nodes will be guided to the closest POI when there are more than one sample hit the target event, as illustrated in Figure 7.

X. Conclusion

In this paper, we discuss the reaction delay problem in the hybrid network. We propose a random deployment and scheduling strategy for the stationary sensor network in order to minimize the detection delay; we also design a number of different deployment strategies as well as motion strategy for mobile sensors. We disclose the relationship between the overall delay bound and different parameters in network planning.

XI. Acknowledgement

The research of authors are partially supported by NSF CNS-0832120, National Natural Science Foundation of China under Grant No.60828003, No.60773042, the Natural Science Foundation of Zhejiang Province under Grant No.$Z1080979$, program for Zhejiang Provincial Key Innovative Research Team, program for Zhejiang Provincial Overseas High-Level Talents (One-hundred Talents Program), National Basic Research Program of China (973 Program) under grant No. $2010CB328100$ and $2010CB334707$, and Tsinghua National Laboratory for Information Science and Technology (TNList). The work is partly supported by China NSF grants (60721002, 60825205) and China 973 project (2006CB303000).

References

[1] Ahn, G.-S., Miluzzo, E., Campbell, A. T., Hong, S. G., and Cuomo, F. Funneling-mac: A localized, sink-oriented mac for boosting fidelity in sensor networks. In *SenSys* (2006).

[2] G. Xing, J. Wang, K. Shen, Q. Huang, X. Jia and H.C. So, Mobility-assisted spatiotemporal detection in wireless sensor networks, In *ICDCS*, 2008.

[3] B. Liu, P. Brass, O. Dousse, P. Nain, and D. Towsley. Mobility improves coverage of sensor networks. In *ACM MobiHoc*, 2005.

[4] T.-L. Chin, P. Ramanathan, and K. K. Saluja. Analytic modeling of detection latency in mobile sensor networks. In *ACM/IEEE IPSN*, 2006.

[5] W. Wang, V. Srinivasan and K.-C. Chua. Trade-offs between mobility and density for coverage in wireless sensor networks. In *ACM MobiCom*, 2007.

[6] N. Bisnik, A. Abouzeid, and V. Isler. Stochastic event capture using mobile sensors subject to a quality metric. In *ACM MobiCom*, 2006.

[7] B. N. Clark, C. J. Colburn, and D. S. Johnson. Unit disks graphs. *Discrete Mathematics*, 86:165C177, 1990.

[8] T. Nieberg, J. Hurink, and W. Kern. A robust PTAS for maximum independent sets in unit disk graphs. In *Proceedings of the 30th workshop on graph theoretic concepts in computer science*, pages 214C221. LNCS 3353, 2004.

[9] D.S. Hochbaum and W. Maass. Approximation schemes for covering and packing problems. *Journal of the ACM*, 32(1):130C136, 1985.

[10] Gandhi, Sorabh and Suri, Subhash and Welzl, Emo Catching elephants with mice: sparse sampling for monitoring sensor networks. *ACM SenSys*, 2007.

[11] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, et al. Aline in the sand: A wireless sensor network for targetdetection, classification, and tracking. *Computer Net-works*, 46(5):605C634, 2004.

[12] G. Tolle, J. Polastre, R. Szewczyk, D. Culler, et al. Amacroscope in the redwoods. In *SenSys*, pages51C63, 2005.

[13] K. Mayer, K. Ellis, and K. Taylor. Cattle health mon-itoring using wireless sensor networks. In *IASTEDCCN*, 2004.

[14] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler,et al. Wireless sensor networks for habitat monitoring.In *WSNA*, pages 88C97, 2002.

[15] W. Xue, Q. Luo, L. Chen, and YunHao Liu. Contour mapmatching for event detection in sensor networks. In *ACM SIGMOD*, 2006.

[16] V. Vapnik and A. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–280, 1971.

[17] GUPTA, P., AND KUMAR, P. R. Critical power for asymptotic connectivity in wireless networks. *Stochastic Analysis, Control, Optimization and Applications: A Volume in Honor of W.H. Fleming, W. M. McEneaney, G. Yin, and Q. Zhang (Eds.)* (1998).

[18] P. Corke, R. Peterson, and D. Rus. Localization and navigation assisted by cooperating networked sensors and robots. *International Journal of Robotics Research*, 24(9), 2005.

[19] Zhaohui Yuan, Rui Tan, Guoliang Xing, Chenyang Lu, Yixin Chen and Jianping Wang Fast Sensor Placement Algorithms for Fusion-Based Target Detection. *IEEE RTSS 08*, 2008.

[20] CARRUTHERS, S., AND KING, V. Connectivity of wireless sensor networks with constant density. In *ADHOC-NOW, LNCS 3158 (2004)*, pp. 149C157.