Efficient Strategies for Collecting Data from Wireless Sensor Network Nodes using Mobile Robots

Onur Tekdas, Nikhil Karnad, and Volkan Isler

Abstract

This work focuses on systems where mobile robots periodically collect data from (static) wireless sensor network nodes. Suppose we are given approximate locations of the static nodes, and an order with which the robot will visit these nodes. We present solutions to the following problems. (i) From the static node's perspective: given the stochastic nature of the robot's arrival, what is an energy-efficient strategy to wake up and send/receive beacon messages? Such a strategy must simultaneously minimize the robot's waiting time and the number of beacon messages. (ii) From the robot's perspective: given the stochastic nature of the wireless link quality, what is an energy-efficient motion strategy to find a good pose (location and orientation) from where the data can be downloaded efficiently? The robot must be able to find such a location quickly but without taking too many measurements so as to conserve the static node's energy.

For the first problem, we present an optimal algorithm based on dynamic programming. For the second problem, we present an efficient, data-driven heuristic based on experiments. Finally, we present a system implementation for an indoor data collection application, and validate our results on this system.

1 Introduction

Wireless sensor networks (WSNs) are finding increasing use in crucial applications such as environmental monitoring, factory automation and security. However, deploying such sensor systems and gathering the data collected by the sensors still remains a challenge. This is especially true when the target application requires collecting data over a large area such as a farm, a forest or a large warehouse.

In cases where the application requires a dense sampling of the environment, the data can be gathered by forming a wireless network where sensor nodes also act as relays. In certain applications, the underlying environment is very large and sampling locations are apart from each other. For example, in some habitat monitoring applications, sensors are deployed to collect humidity and temperature data across the entire habitat of species [11]. In building automation, a WSN can be rapidly deployed to collect data (temperature, light) in a few key locations in a large warehouse.

Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN - 55414, USA

In these applications, a deployment that is dense enough to form a connected network can be costly and difficulty to maintain. It is also very expensive to install a wired network just for data acquisition. Often, instead of deploying a dense, connected network, data is manually downloaded from the motes.

An alternative to collecting data manually is to use mobile robots. The last decade witnessed significant developments in mobile robot navigation. It is now feasible to develop systems where robots periodically visit sensor nodes and gather the data collected by them. Typically mobile robots are capable of carrying large batteries and are easily rechargeable. Therefore, the life-time of the static nodes is usually the most crucial factor in determining the overall life-time of the system. In this work, we focus on such systems and address two problems that affect the life time of the automation system.

Beacon Scheduling: In most applications, a robot's arrival to a sensor's vicinity will be a stochastic process due to uncertainties in the navigation times of robots and changes in their trajectories. Therefore, sensor nodes must wake-up, send beacon messages and listen to the channel for availability of robots. If this is done very frequently, energy consumed in beaconing can reduce the life-time of the network. In Section 3, we address the problem of scheduling beacon messages and present optimal beaconing algorithms.

Data Download: The quality of the communication link between a robot and a sensor can affect the time to download the data (and hence the energy consumption) drastically. If the robot can utilize its mobility and find a "good" location to download data, this can yield significant energy savings. This statement is further justified in Section 4 where we address the search problem and present a data-driven strategy to find a good download location. In indoor environments where the behavior of the signal is unpredictable due to multipath effects and the dynamic nature of the environment, it is easy to see that there is no online algorithm with provable performance guarantees[1]. Therefore, we present a heuristic strategy based on extensive experiments we performed to understand the effect of robot's location and orientation on the signal quality.

In Section 5, we present the details of a system implementation that utilizes robots for gathering data, and demonstrate the utility of our algorithms with experiments run on this system. We start with an overview of related work.

2 Related Work

Mobility in collecting sensor data is extensively studied. For example, in [13], Shah et al presented an architecture that uses mobile entities in the environment for data delivery. In most of the related literature, mobility is treated as an uncontrolled process.

More recently, researchers proposed architectures that exploit controlled mobility [14, 6, 3, 18, 15]. A recent review on the state of the art in exploiting sink mobility can be found in [9].

[1] For any given online strategy, an adversary can pick the "good" location to be the last location visited by the online algorithm.

In existing approaches, the robot's trajectory is either given or computed in advance. This constraint is relaxed in [6] where the robot's velocity is modified (along a fixed path) to improve transmission quality. In more recent work, a system where robots efficiently collect data from static sensors have been presented [17]. This work demonstrates the energy savings attained by using mobile robots over using static relay nodes. In the present work, we take a further step and present a strategy that fully utilizes the controllability of the robot's mobility (position and orientation) to find good download locations in an online fashion. The strategy does not make strong assumptions about the wireless signal, and we show its utility with real implementations.

We also study the interactions between the robot and static nodes during the discovery phase. This is related to sleep scheduling which is usually studied as a topology management problem [2, 12]. In this work, we focus on a special case where the arrival of the robot is given as a probability distribution, and compute the optimal sleep schedule which simultaneously minimizes the number of beacon messages and the robot's wait time. In a real system implementation, we show how such distributions can be learned over time.

The interactions between robots and a static sensor network have also been addressed in the robotics literature for network repair [4], connectivity [1] and navigation [7] problems. In this work, we model the interaction between the robot and the nodes at a lower level and address signal-strength and node scheduling issues. Such approaches have recently started appearing in the robotics literature, mostly in tracking [8, 10] and connectivity maintenance [5, 16] applications.

3 Optimal beacon scheduling

A robot's arrival to a sensor mote's vicinity is a stochastic process, due to uncertainties in navigation. It is thus necessary for the mote to periodically send beacon messages and execute a receiver check to establish connection with the robot[2]. By adapting the beacon interval to match the robot's arrival pattern, we can keep the duty cycle of the mote as low as possible, thereby conserving energy and increasing its life-time.

3.1 Formulation

Consider a system in which m sensor motes have been statically deployed. The approximate location of each mote is known w.r.t. a fixed origin: the home base from where the robot starts its journey. An ordered sequence of labels $S = \{s_1, s_2, \ldots, s_m\}$ is assigned to the sensor motes a priori.

Although the path of the robot is specified, there are many sources of uncertainty that contribute to the robot's arrival at each sensor being a stochastic process. For instance, the robot may spend uncertain amounts of time to locate the motes and

[2] In general, listening to the channel and sending beacon messages can be decoupled. In this case, the discussion can be modified to focus on the most energy expensive portion (typically receiving messages).

download data from them. It may take alternate routes due to obstacles, and spend time to compensate for uncertainties in its own actuators and sensors.

Ideally, if the mote knew the exact time (or the interval) the robot would be within its communication range, then only one beacon round would suffice. A practical alternative is to send beacon messages infrequently and have the robot wait in the mote's vicinity until it hears a beacon. This has the undesired effect of making the time to visit all nodes large which, in turn, decreases the overall system performance.

The time interval between consecutive robot arrivals at sensor mote s is called the *interarrival time*. Due to the stochastic nature, we represent the robot's interarrival time at sensor s as a probability distribution. This distribution can be either guessed (e.g. by using a Gaussian which represents the expected arrival time and uncertainty), or can be adaptively learned by keeping the history. In this work, we assume that the distribution is given and omit the details of how it can be learned due to space limitations.

We assume that the time between consecutive robot arrivals at a mote is bounded from above. In general, there can also be a lower bound that is non-zero, because the robot's velocity and the length of the complete path ensure that the robot takes a non-zero amount of time to revisit the same sensor.

We now solve the following problem: Given a robot interarrival probability distribution at a mote, find a beacon schedule for that mote, using the least number of beacons possible, such that the expected waiting time of the robot between its arrival at that mote, and receiving a beacon message is bounded from above by a predetermined value T_w. In Section 3.2.1, we show how such a beacon schedule can be computed.

3.2 Optimal Solution

We now focus on the scenario where the mobile robot visits a sensor s in rounds. As explained in the formulation, we assume that for any round, the interarrival time is bounded: there is a time before which the robot cannot be present in communication range of the mote, and after which the robot is guaranteed to have visited the mote. Denote this interval as T. We model time as discrete by dividing the interval T into n time instants spaced equally by Δt units, $T = \{t_1, \ldots, t_n\}$. A beacon can be scheduled at any t_i ($1 \leq i \leq n$).

The robot's arrival during the interval T is given as a probability distribution over the time instants in T. Let $p(t_i)$ denote the probability that the robot arrives at time t_i. In the case that the probability is continuous over time, one can interpret t_i to be the end point of a time interval $[t_{i-1}, t_i]$, with the beacon being placed at the end of that interval and $p(t_i)$ being the aggregate probability for that interval.

Let $B = \{b_1, \ldots, b_k = t_n\}$ be a beacon schedule. The values b_i denote the times in the interval T at which beacons are scheduled. Given the arrival distribution p, the robot's expected waiting time $ET(B, p)$ at sensor mote s is given by

$$ET(B,p) = \sum_{i=0}^{k-1} \sum_{t_j = b_i + \Delta t}^{b_{i+1}} p(t_j)(b_{i+1} - t_j) \tag{1}$$

In Equation 1, we define b_0 as the start time of T.

Consider beacon b_i. For any robot arrival at time $t_j > b_i$, the expression $p(t_j)(b_{i+1} - t_j)$ is the expected waiting time for the robot, until beacon b_{i+1} is heard. Thus the inner summation in (1) gives the robot waiting times between beacons b_i and b_{i+1}. The outer summation accumulates the expected waiting times over the entire schedule.

We would like to simultaneously minimize the cardinality of B and the expected waiting time of the robot. Let the cardinality of B be denoted by $k = |B|$.

For a given value of k, we formulate the following decision problem: Given parameters k, T_w and an arrival distribution p, can we find a beacon schedule B such that $|B| = k$ and $ET(B,p) \leq T_w$? To minimize the number of beacons used to satisfy this T_w, we perform a search over the possible values of k by solving the decision problem for each value.

Typically, the time taken by the robot to traverse the whole round is much larger than the length of T. For such cases, we obtain the following insight about an optimal schedule.

Lemma 1. *During each round, there has to be a beacon at the last instant of time in T, to ensure that $ET(B,p) \leq T_w$.*

Proof. We prove this claim by contradiction. Suppose there is no beacon at the last time instant over which $p(t_i)$ is distributed. Let $t_j < t_n$ be the time at which the last beacon is scheduled. If the robot arrives after t_j, but before t_n, then it has to wait until the mote's next beacon, which, according to the distribution occurs only at the estimate of the next robot interarrival time. Since this can be of the order of the duration of a round, the waiting time can grow arbitrarily large, giving us a value greater than T_w: a contradiction.

Lemma 1 gives us a starting point to place a beacon: at time t_n (the end of T). Also, note that, in Equation 1 the robot's waiting time is determined by only the first beacon that it hears after arrival. This motivates the following dynamic programming solution.

3.2.1 Dynamic programming

Let the robot interarrival distribution for mote s be p, distributed over the interval T. Let k be the number of beacons to be scheduled in the time interval T. We seek to answer the question: What is the minimum value of k such that the beacons satisfy the constraint on expected waiting time T_w?

Let the cost function be $C(i,t_j)$ which denotes the expected waiting time ("cost") for robot arrivals *after* interarrival time t_j, when beacon i is scheduled at time t_j.

$$C(i,t_j) = \min_{t_j < t_r < t_n} \left\{ \sum_{t_q=1+t_j}^{t_r} p(t_q) \cdot (t_r - t_q) + C(i+1,t_r) \right\} \tag{2}$$

Since beacon i is placed at time t_j and beacon $i+1$ at time t_r, the first term on the right-hand side in Equation 2 computes expected waiting time for a robot arrival

between those two beacons. The second term computes the expected waiting time for arrivals after time t_r.

Since there must be a beacon at t_n (see Lemma 1), $C(k,t_n) = 0$. Further, we do not allow beacon k to be scheduled anywhere except at time t_n, thus $C(k,t_y) = \infty$ for $t_1 \leq t_y < t_n$. We then use Equation 2 to compute the rest of the cost table, which in total is of size $k \times n$. To complete the computation, we need to increment each value $C(1,t) \forall t$ by the expected waiting time for robot arrivals from t_0 to t. This accounts for robot arrivals *before* the first beacon.

The time at which the first beacon should be scheduled is: t_j such that the value of $C(1,t_j)$ is minimum i.e. $b_1 = \arg\min_{t_j} C(1,t_j)$. Since the computation of $C(1,t_j)$ used a minimum value for some $C(2,t_r)$, we backtrack to find the best possible scheduling times $b_2, b_3, \ldots, b_k$.

We want the value of k to be the least possible to satisfy the expected waiting time constraint. In order to do this minimization, we start with just $k = 1$ beacon i.e. a cost table of size $1 \times n$ and increase k if the expected waiting time for that k exceeds T_w. Instead of this linear search, we could also use a binary search to find the best k, but that approach does not allow us to incrementally build the cost table. As a result, it ends up computing the whole table and then eliminating half of the values at each step. The entire computation can be performed in $O(kn^3)$ steps.

3.2.2 Simulation results

We demonstrate the utility of using the dynamic programming algorithm through simulated robot arrival times and beacon schedules.

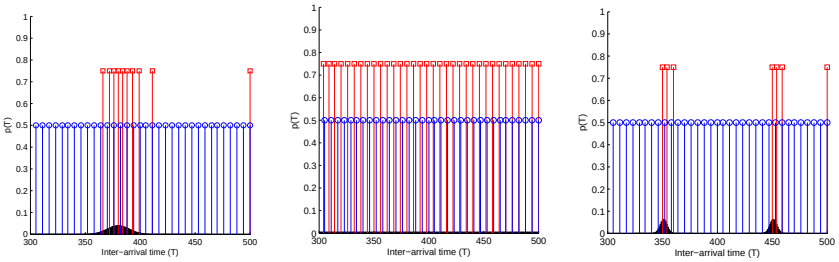

Fig. 1 A comparison of optimal beacon scheduling (red circles, tall) to uniform scheduling (blue circles, short) for different robot arrival patterns: unimodal Gaussian (left), uniform (middle), and bimodal Gaussian (right).

We compare three different types of robot interarrival patterns: uniform, Gaussian and bimodal Gaussian (mixture of two Gaussians). In all three cases, the desired waiting time is 2.5 seconds and the robot interarrival times lie between 300 sec and 500 sec.

Figure 1 (left) models the robot's arrival pattern at a mote as a unimodal Gaussian. Uniform beaconing uses 34 beacons for an expected waiting time of 2.489 sec-

onds. In contrast, our algorithm uses 10 beacons ($\approx$ 70% better) with an expected waiting time of 2.233 seconds.

The algorithm is applicable to any type of arrival distribution. For instance, if the arrival pattern at a mote is a bimodal Gaussian (right of Fig. 1 $\mathcal{N}$(350,10) and $\mathcal{N}$(450,10) with equal weights), our solution uses 7 beacons ($\approx$ 79% better) with an expected wait time of 2.115 seconds.

4 Local search

When the robot is downloading data from a node, the quality of the wireless communication link is a crucial factor in determining the life time of the node: when the link quality is high, the same amount of data can be transferred using less energy. In this section, we present a motion strategy for a robot to find a good location to download the data. The algorithm is based on insights from a series of experiments which we describe next.

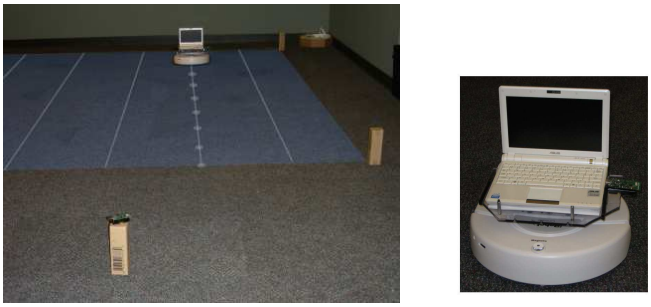

Fig. 2 Left: Experimental setup to measure the link quality of data transfer from a mobile robot to a base station, with the robot moving on a uniform grid.

We started our experiments by collecting data using the setup shown in Figure 2 where we placed an 11×11 grid on a $3m \times 3m$ indoor environment. We mounted a base station mote on our robot (iRobot Create with Asus Eee PC) and the robot autonomously visited grid points while pointing to a fixed direction.

In the first experiment, we placed a data node at location (3,10) in Figure 3. The robot visited each location, and took 50 measurements. Each measurement was taken by sending a 4 byte message during which RSSI (radio signal strength indicator) and LQI (link quality indicator) values were recorded. The left plots in Figure 3 show the mean (top) and median (bottom) values of the LQI measurements. The right plots show the RSSI values. As it can be seen in the figure, all plots give a unified view of the link quality. The main observation from this experiment is that although in general the LQI increases as we get closer to the sensor node, the surface is not smooth and contains many deep drops due to multi-path effects.

The next experiment illustrates the effect of link quality on the time to download the data. In Figure 4, the top figure shows the time to download 50 messages from each grid point. The peaks show a correlation with the deep fading affects in the

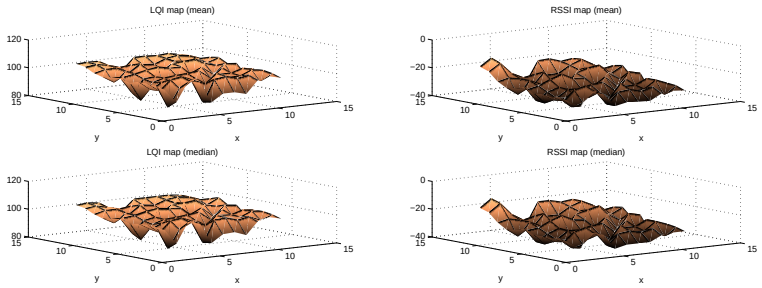

Fig. 3 The robot visited each location, and took 50 measurements. **Left:** mean (top) and median (bottom) values of the LQI measurements. **Right:** mean (top) and median (bottom) values of the RSSI measurements.

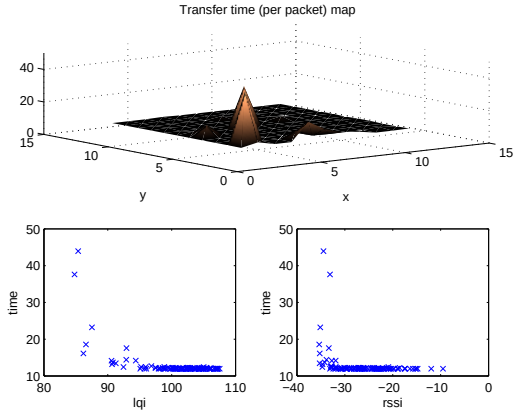

Fig. 4 Time to download 50 messages from each grid point as function of location (top), and as functions of LQI (bottom-left) and RSSI (bottom-right). With controlled mobility, the robot can decrease the download time significantly by moving slightly.

previous experiment (Figure 3). Bottom left (resp. right) figure shows the relation between LQI (resp. RSSI) measurements and time transfer. As it can be seen in the left figure, the transfer time is very low for LQI measurements above 95. On the other hand, the transfer time increases drastically for values lower than 95. *This observation shows the potential utility of controlled mobility: robots can reduce the data download time (and increase the life of the sensor network) by finding a "good"*[3] *location to download the data.*

The next experiment sheds further light on path-loss and multi-path affects on link quality. To cover a wider range, we moved the robot on a line-segment in a corridor in our building and placed a mote on the mid-point of this line segment. In Figure 5-left, the mote is located at $x = 26$ and robot starts taking measurements

[3] above 95 in this case.

at $x = 1$ and ends at $x = 51$. The discretization level is 1 foot and robot takes 50 measurements from each location. Top figure shows the mean values of 50 measurements and bottom figure shows the median of the measurements. As expected, the link quality increases when robots get closer to the sensory mote, while it tends to decrease while getting further away from the sensory mote. After performing similar experiments, we concluded that the following observation explains the link-quality behavior better: *Within a certain range (± 8 ft, in this case), the link quality is consistently "good" and unpredictable (random) outside this range.*

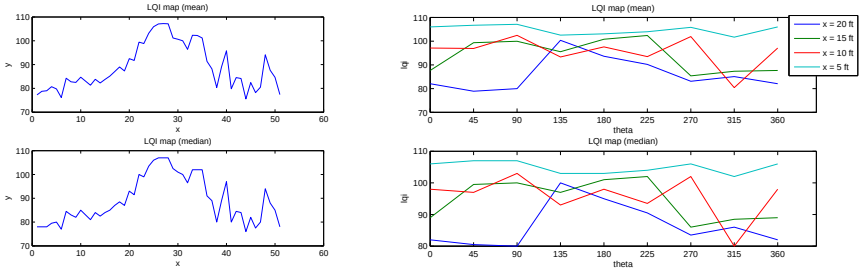

Fig. 5 Left: The mote is located at $x = 26$ and the robot moves from $x = 1$ to $x = 51$. Within a range of ± 8 ft, the link quality is consistently "good". It is unpredictable (random) outside this range. **Right:** The θ-values correspond to robot orientations. Each curve corresponds to a different location on a line. The sensor is located at $x = 0$. The behavior of RSSI or LQI as a function of rotation is not easily predictable.

Most robotic systems have a rotational component which means that we can control the orientation of the robot. Therefore, the robot should search for not only a good location but a good orientation as well. The next experiment focuses on this aspect. Figure 5-right shows the change in link quality with the various orientations of the base station (on the robot) at fixed locations. The figure shows the results for 4 fixed points at distances 5,10,15 and 20 feet from the mote. The results show that when the base station is close to the sensor mote, the orientation does not affect the link quality significantly. However, when the distance is large, small changes in orientation may result in drastic changes in link quality. Moreover, this change is not easily predictable. For example, when the robot is at the furthest point (the 20-feet curve), sensor mote and base station point towards each other when the angle is 180°. In this orientation the LQI is 95. If robot turns 45° in counter-clockwise direction, the LQI value increases to 100. If the robot turns 45° more on counter-clockwise direction, then the LQI value suddenly drops to 80. This example also shows that measurements from various orientations may not give a clear indication about the direction of the sensory mote.

The results of these experiments can be summarized as follows:

- Within a certain distance (an environment dependent parameter), the signal quality is predictably good and the orientation of the robot does not make a significant difference.

- When the robot is outside this range, it is very difficult to use local information (such as gradient) to find the location or orientation of the sensor.

In the next section, we present a search strategy based on these observations.

4.1 The search algorithm

In this section, we describe a search algorithm to find a good download location. In many applications, it is beneficial to search for this location in an online fashion because the location of the mote whose data will be downloaded can change locally, the signal properties may change over-time, or the robot may not have the localization capability to visit a location accurately.

As mentioned in the previous section, it is very difficult, if not impossible, to use local gradient information to seek a good location. A more global approach is needed. The strategy we present uses two environment dependent parameters. The first parameter β is a lower-bound for an acceptable signal strength (LQI value). For example, an appropriate β value for the environment where the experiment shown in Figure 4 was performed, is 95. The second value α is mainly a grid resolution and it is set to the distance within which the link quality is predictably good. For the environment where the experiment shown in Figure 5-left was performed, an appropriate α value is 8 ft.

Upon hearing a beacon message, the robot finds a good location by placing a grid on the environment where the dimension of each cell is determined by α. When the robot visits a grid cell, it rotates $0, 90, 180, 270$ degrees. This allows us to get rid of local multi-path effects and to simultaneously seek a good orientation. At each rotation, the robot takes five link quality measurements. The quality of each orientation is defined as the median of these five measurements. The weight of each cell is then set to the the highest of these four median values. In the algorithm below, $measure(c)$ subroutine performs these steps at cell location c. We also keep track of a table where we store the expected link qualities. For each unvisited cell, we set the average of neighbor cells which are visited before as the *expected_weight* value for that cell. Next, robot visits the location with maximum expected weight.

The robot searches for a good grid cell using the following heuristic:

Algorithm 1 *LocalSearch*

1: $expected\_weight(c \in C) \leftarrow 0$, $c \leftarrow (0,0)$ *(initial location)*
2: ***while*** *there are unexplored cells* ***do***
3: ***if*** $measure(c) \geq \beta$ ***then***
4: ***return***
5: ***end if***
6: *Forall* $c' \in Neighbor(c)$ *if it is unvisited, make update:*
$expected\_weight(c') = mean(\forall_{c'' \in Neighbor(c')} measure(c''))$
7: $c \leftarrow \max_{c'} expected\_weight(c')$
8: ***end while***

A couple of comments are in order. If the β value is not known, we can set it to a high value. In this case, the robot will visit all grid-cells. We can then pick the best

location. Second, it is very easy to incorporate collision avoidance into the strategy by setting the weight of a cell to zero if there is an obstacle at that cell.

In the next section, we demonstrate the utility of this strategy with a series of experiments.

4.2 Search experiments

We tested the search algorithm in a number of settings. In this section, we present two of these results.

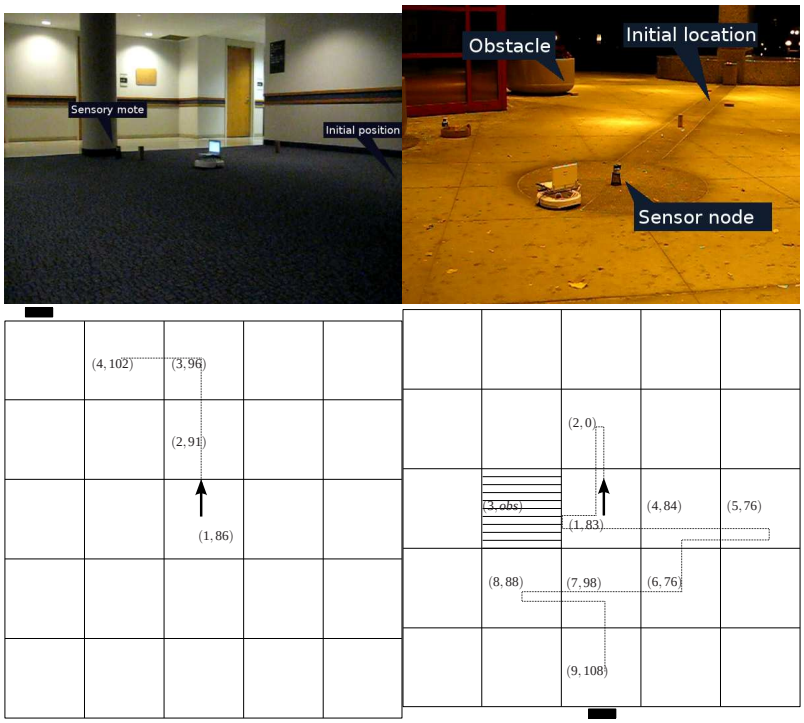

Fig. 6 Bottom figures show a virtual grid used by the search algorithm. The black rectangles show the location of the mote. **Top Left:** The setup for an indoor experiment. The picture shows the best configuration found by the search algorithm. **Bottom Left:** Steps in finding a good location in the setup shown on top. **Top Right:** Search performed in an outdoor setting. The picture shows the best configuration found by the search algorithm. The shaded cell corresponds to the obstacle that robot avoided. **Bottom Right:** Steps taken during outdoor search.

The first experiment was performed in the indoor setting shown in Figure 6-left. The signal strength in TelosB mote was set to 3. The other system parameters were $\alpha = 1.5m$ and $\beta = 100$.

When the robot started from the initial location shown in the figure, it quickly converged to a good location. Robot's steps are shown in Figure 6-left bottom where

the visited cells are labeled with the format (s,r): s is the order the cell was visited and r is the maximum value sampled from four orientations.

The second experiment was performed in an outdoor setting with $\alpha = 3m$ and $\beta = 100$. As shown in Figure 6-right, the robot quickly converged to a good location.

In conclusion, the simple search strategy presented in this section was very efficient in finding a good download location. Where most local search heuristics would get stuck with a single cell, the presented search strategy quickly converges to a robot pose from where the data can be downloaded efficiently.

5 System design

In this section, we describe a system which incorporates the results presented in this paper. In Section 5.3 we present experimental results which demonstrate the utility of these components.

5.1 Hardware Components

Our system consists of three classes of devices. (i) The sensor motes are Crossbow Telos, (rev. B) which use the CC2420 chip. They are IEEE 802.15.4 compliant. We deployed three static motes in the fourth floor lounge of the Digital Technology Center (DTC, Walter Library) at the University of Minnesota, Twin-Cities. The experimental setup is shown in Figure 7. (ii) The mobile robot is an iRobot Create without the command module. (iii) The control program for the robot runs on an Asus Eee PC, which interfaces with the Create directly through a USB-to-Serial cable. The system ran Linux (Ubuntu) and our Java and C++ programs used serial communication libraries to write motion commands to the robot, in accordance with the Create Open Interface (OI) specifications.

5.2 Adaptive Beacon Scheduling

The control program on the TelosB motes was written in the `nesC` language, then compiled and programmed onto the mote using `TinyOS 2.x`. In our design, sensing motes transmit beacon messages and the base station mote attached to the mobile robot listens for these messages.

To allow the TelosB motes to have an adaptive beacon schedule, we store a beacon time interval array on each mote. A one-shot timer cycles through the array, allowing the mote to keep its transmitter off for arbitrary intervals. We set the receiver sleep interval using the `LowPowerListening` interface. However, we believe that since we have packet acknowledgments enabled, the receiver of the sensing mote is turned on every time it sends a beacon. This design decision could be replaced with unacknowledged packets. In both cases, our optimal beacon schedule helps save power on the mote by reducing the amount of time during which the transmitter and/or the receiver are active.

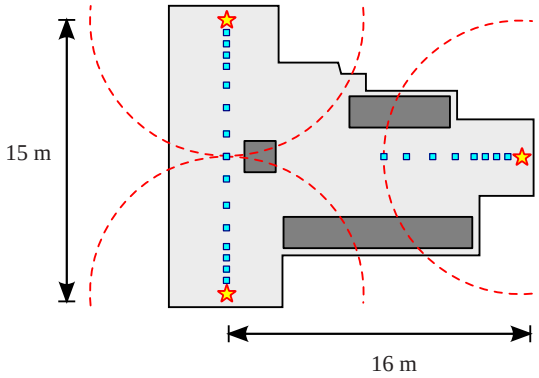

Fig. 7 A proof-of-concept deployment. The stars are approximate locations of the data nodes. The dashed lines show their communication range. The squares are locations where the robot starts either the download or the local search.

5.3 Experiments

We performed four experiments to demonstrate the utility of incorporating both beacon scheduling, and local search. The experiments are: baseline (B), beacon scheduling (BS), local search (LS) and both local search and beacon scheduling (LSBS). Each experiment consisted of 8 rounds. In each round, robot visits a predefined location for each mote, and downloads the data from that mote. The locations that the robot starts downloading (shown as squares in Fig. 7) are fixed for comparison purposes: For example, if in experiment B the robot downloads from a fixed location then in experiment LS, the robot starts the local search from the same location.

We picked a range of download locations in a mote's vicinity to simulate the effects of localization uncertainty: If the robot does not have accurate means of localization, even if it targets a fixed location to download data, it may be off from that location by a distance given by the uncertainty range. After arriving at a predetermined location, the robot either directly downloads the data (experiment B and BS), or performs a local search to find a good location before downloading (LS and LSBS experiments). After download finishes, the robot either continues to the next mote directly (experiments B and LS), or computes an updated beacon schedule based on interarrival times, uploads it to that mote and proceeds to the next mote (experiments BS and LSBS).

In all experiments, the beacons are special messages whose payload consists of (i) the node id of the mote, and (ii) a sequence number of the triggered beacon. To compare the local search with base case, we needed a mechanism to compare the trade-off between energy gain in efficient download and energy spent in extra beacons sent during the search phase. Therefore, we used data packages which are the same as the beacon type messages. To download the data on the mote, the robot must successfully hear 100 additional beacons. This represents scenarios where the data stored on the mote corresponds to 100 messages and all of it must be success-

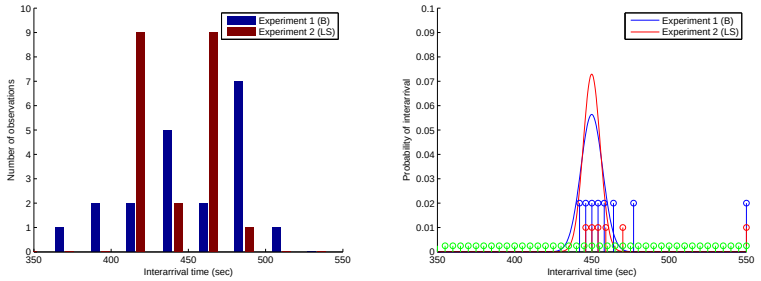

Fig. 8 The robot interarrival times from our experiments were modeled as normal distributions.

fully downloaded. This way, we can use the last received beacon sequence number for each mote to represent the total energy consumption metric spent in beaconing and download. Even with this modest amount of data, each experiment lasted about an hour.

In experiment B, we choose the beacon interval for discovery phase as 5 seconds. This guarantees an expected robot waiting time of 2.5 sec. The optimal beacon scheduling algorithm of Section 3 and the robot inter-arrival distribution observed in experiment B are used to achieve 2.5 sec waiting time in experiment BS. In experiment LSBS, we used the interarrival times from the LS experiment to compute the optimal beacon schedule for this case. The recorded interarrival times and the robot's arrival model are shown in Figure 8. In experiment BS, the optimum beacon schedule uses 8 beacons.

In comparison to the number of beacons ($550/5 = 110$) in the base experiment B, the beaconing strategy yields significant energy savings (8 beacons) while satisfying the same expected waiting time constraint. Comparing the total number of beacons, we can see the effect in total performance. Beacon scheduling in experiment BS reduced the total number of beacons to 2791 compared to 4839 in experiment B, the baseline (first and second columns in right of Table 1).

The left one of the two tables shown in Table 1, shows the packet loss rates for various locations in each experiment. Clearly, local search provides a significant reduction in packet loss rate for the first two locations (compare B versus LS and BS versus LSBS) where the distance prevents a lossless communication between the mote and robot. For example, in the experiment B, the first mote has to sent 538 packets until the robot successfully downloads all of the 100 data packets, whereas after local search no packet is lost. We can see the efficiency of local search for the first two rounds of the examples in table on the right (Table 1). On the other hand, for the rest of the rounds, the local search does not provide significant gains. In fact, the energy consumption slightly increases in experiment LSBS compared to BS experiment due to the overhead (i.e. number of beacons sent during local search).

It is worth noting that in this indoor setting, the robot's total path is relatively short compared to the search distance. Thus, the search overhead becomes compa-

rable to the number of discovery beacons. When the travel distances are large, (e.g. in outdoor settings), the search overhead will become negligible. In this case, local search will yield more significant energy savings.

Dist.	B	BS	LS	LSBS
7.5m	173%	36%	0%	1%
6.25m	7%	21%	0%	1%
5m	11%	0%	0%	0%
3.75m	8%	0%	0%	0%
2.5m	2%	3%	0%	3%
1.7m	0%	0%	0%	0%
0.9m	0%	0%	0%	0%
0.1m	0%	0%	0%	0%

	B	BS	LS	LSBS
1-2	1642	897	997	828
3-8	3197	1894	3215	2152
Total	4839	2791	4212	2980

Table 1 Left table shows the package loss rates for each experiments (B:Base,LS:Local Search,BS: Beacon Schedule, LSBS: Local Search and Beacon Schedule together) with respect to the distance that robot starts to download or starts to the local search. For each download we calculate the number of packet loss until robot hears 100 beacons. **Right** figure shows the total number of beacons send from 3 motes during the experiment.

Overall, the experiments clearly demonstrate that (i) adaptive beaconing strategies yield significant savings in the number of discovery beacons sent, and (ii) local search strategies can result in drastic improvements in the download time when the link quality is unpredictable.

6 Conclusion

In this paper, we addressed two problems that arise in applications where robots collect data from static nodes. In the first problem, the goal is to minimize the energy spent by the static nodes for beaconing. For this problem, an optimal beaconing strategy based on dynamic programming was presented. In the second problem, the goal is to minimize the energy spent in communication. For this purpose, we present a strategy for the robot to adaptively discover a download location where the signal is strong. The strategy is based on insights gathered by experiments. We report these in the paper as well. Finally, we present an indoor system for data collection which incorporates the algorithms presented in the paper. Experiments performed on the system demonstrate the utility of the two results in the paper.

There are additional factors (e.g. robot's interarrival times, the amount of data to be downloaded at each round) which effect the overall system performance. Currently, we are building an outdoor system for habitat monitoring including a new robotic platform. In the near future, we will demonstrate the use of these results within the context of a field application in environmental monitoring.

Acknowledgments

This work is supported in part by NSF Grants 0907658, 0917676 and 0707939.

References

1. Atay, N., Bayazit, B.: Mobile wireless sensor network connectivity repair with k-redundancy. In: Proceedings of the 2008 International Workshop on the Algorithmic Foundations of Robotics, WAFR (2008)
2. Cerpa, A., Estrin, D.: ASCENT: Adaptive Self-Configuring sEnsor Networks Topologies. IEEE Transactions on Mobile Computing **3**(3), 272–285 (2004)
3. Chatzigiannakis, I., Kinalis, A., Nikoletseas, S.: Efficient data propagation strategies in wireless sensor networks using a single mobile sink. Computer Communications **31**(5), 896–914 (2008)
4. Corke, P., Hrabar, S., Peterson, R., Rus, D., Saripalli, S., Sukhatme, G.: Autonomous deployment and repair of a sensor network using an unmanned aerial vehicle. In: Proceedings of the 2004 IEEE International Conference on Robotics and Automation (ICRA), vol. 4, pp. 3602–3608 (2004)
5. Dixon, C., Frew, E.: Controlling the mobility of network nodes using decentralized extremum seeking. Decision and Control, 2006 45th IEEE Conference on (2006)
6. Kansal, A., Somasundara, A.A., Jea, D.D., Srivastava, M.B., Estrin, D.: Intelligent fluid infrastructure for embedded networks. In: MobiSys '04, pp. 111–124. ACM (2004)
7. Li, Q., Rus, D.: Navigation protocols in sensor networks. ACM Transactions on Sensor Networks **1**(1), 3–35 (2005)
8. Lindhe, M., Johansson, K.: Communication-aware trajectory tracking. Robotics and Automation, 2008. ICRA 2008 pp. 1519–1524 (2008)
9. Ma, J., Chen, C., Salomaa, J.P.: mWSN for Large Scale Mobile Sensing. Journal on Signal Processing Systems **51**(2), 195–206 (2008)
10. Mostofi, Y.: Communication-aware motion planning in fading environments. Robotics and Automation, 2008. ICRA 2008 pp. 3169–3174 (2008)
11. Musăloiu-E., R., Terzis, A., Szlavecz, K., Szalay, A., Cogan, J., Gray, J.: Life Under your Feet: A Wireless Sensor Network for Soil Ecology. In: EmNets Workshop (2006)
12. Schurgers, C., Tsiatsis, V., Ganeriwal, S., Srivastava, M.: Topology management for sensor networks: exploiting latency and density. In: MobiHoc, pp. 135–145 (2002)
13. Shah, R.C., Roy, S., Jain, S., Brunette, W.: Data mules: modeling and analysis of a three-tier architecture for sparse sensor networks. Ad Hoc Networks **1**(2–3), 215–233 (2003)
14. Somasundara, A.A., Kansal, A., Jea, D.D., Estrin, D., Srivastava, M.B.: Controllably mobile infrastructure for low energy embedded networks. IEEE Transactions on Mobile Computing **5**(8), 958–973 (2006)
15. Tariq, M.M.B., Ammar, M., Zegura, E.: Message ferry route design for sparse ad hoc networks with mobile nodes. In: MobiHoc '06, pp. 37–48 (2006)
16. Tekdas, O., Isler, V.: Robotic routers. In: Proceedings of the 2008 IEEE International Robotics and Automation. ICRA 2008, pp. 1513–1518 (2008)
17. Tekdas, O., Lim, J., Terzis, A., Isler, V.: Using mobile robots to harvest data from sensor fields. IEEE Wireless Communications (2008)
18. Wang, W., Srinivasan, V., Chua, K.C.: Using mobile relays to prolong the lifetime of wireless sensor networks. In: MobiCom '05, pp. 270–283. ACM (2005)