Mobile Relay Configuration in Data-intensive Wireless Sensor Networks

Fatme El-Moukaddem, Eric Torng, Guoliang Xing

Abstract— Wireless Sensor Networks (WSNs) are increasingly used in data-intensive applications such as micro-climate monitoring, precision agriculture, and audio/video surveillance. A key challenge faced by data-intensive WSNs is to transmit all the data generated within an application's lifetime to the base station despite the fact that sensor nodes have limited power supplies. We propose using low-cost disposable mobile relays to reduce the energy consumption of data-intensive WSNs. Our approach differs from previous work in two main aspects. First, it does not require complex motion planning of mobile nodes, so it can be implemented on a number of low-cost mobile sensor platforms. Second, we integrate the energy consumption due to both mobility and wireless transmissions into a holistic optimization framework. Our framework consists of three main algorithms. The first algorithm computes an optimal routing tree assuming no nodes can move. The second algorithm improves the topology of the routing tree by greedily adding new nodes exploiting mobility of the newly added nodes. The third algorithm improves the routing tree by relocating its nodes without changing its topology. This iterative algorithm converges on the optimal position for each node given the constraint that the routing tree topology does not change. We present efficient distributed implementations for each algorithm that require only limited, localized synchronization. Because we do not necessarily compute an optimal topology, our final routing tree is not necessarily optimal. However, our simulation results show that our algorithms significantly outperform the best existing solutions.

Index Terms—Wireless sensor networks, energy optimization, mobile nodes, wireless routing

1 Introduction

WSNs have been deployed in a variety of *data-intensive* applications including micro-climate and habitat monitoring [1], precision agriculture, and audio/video surveillance [2]. A moderate-size WSN can gather up to 1 Gb/year from a biological habitat [3]. Due to the limited storage capacity of sensor nodes, most data must be transmitted to the base station for archiving and analysis. However, sensor nodes must operate on limited power supplies such as batteries or small solar panels. Therefore, a key challenge faced by data-intensive WSNs is to minimize the energy consumption of sensor nodes so that all the data generated within the lifetime of the application can be transmitted to the base station.

Several different approaches have been proposed to significantly reduce the energy cost of WSNs by using the mobility of nodes. A robotic unit may move around the network and collect data from static nodes through one-hop or multi-hop transmissions [4], [5], [6], [7], [8]. The mobile node may serve as the base station or a "data mule" that transports data between static nodes and the base station [9], [10], [11]. Mobile nodes may also be used as *relays* [12] that forward data from source nodes to the base station. Several movement strategies for mobile relays have been studied in [12], [13].

Although the effectiveness of mobility in energy conservation is demonstrated by previous studies, the following key issues have not been collectively addressed. First, the movement cost of mobile nodes is not accounted for in the total network energy consumption. Instead, mobile nodes are often assumed to have replenishable energy supplies [7] which is not always feasible due to the constraints of the physical environment. Second, complex motion planning of mobile nodes is often assumed in existing solutions which introduces significant design complexity and manufacturing costs. In [7], [8], [14], [15], mobile nodes need to repeatedly compute optimal motion paths and change their location, their orientation and/or speed of movement. Such capabilities are usually not supported by existing low-cost mobile sensor platforms. For instance, Robomote [16] nodes are designed using 8-bit CPUs and small batteries that only last for about 25 minutes in full motion.

In this paper, we use low-cost disposable mobile relays to reduce the total energy consumption of data-intensive WSNs. Different from mobile base station or data mules, mobile relays do not transport data; instead, they move to different locations and then remain stationary to forward data along the paths from the sources to the base station. Thus, the communication delays can be significantly reduced compared with using mobile sinks or data mules. Moreover, each mobile node performs a single relocation unlike other approaches which require repeated relocations.

Our approach is motivated by the current state of mobile sensor platform technology. On the one hand, numerous low-cost mobile sensor prototypes such as Robomote [16], Khepera [17], and FIRA [18] are now

• *Fatme El-Moukaddem, Eric Torng, and Guoliang Xing are with the Department of Computer Science, Michigan State University.*

available. Their manufacturing cost is comparable to that of typical static sensor platforms. As a result, they can be massively deployed in a network and used in a disposable manner. Our approach takes advantage of this capability by assuming that we have a large number of mobile relay nodes. On the other hand, due to low manufacturing cost, existing mobile sensor platforms are typically powered by batteries and only capable of limited mobility. Consistent with this constraint, our approach only requires one-shot relocation to designated positions after deployment. Compared with our approach, existing mobility approaches typically assume a small number of powerful mobile nodes, which does not exploit the availability of many low-cost mobile nodes.

We make the following contributions in this paper. (1) We formulate the problem of *Optimal Mobile Relay Configuration* (OMRC) in data-intensive WSNs. Our objective of energy conservation is *holistic* in that the total energy consumed by both mobility of relays and wireless transmissions is minimized, which is in contrast to existing mobility approaches that only minimize the transmission energy consumption. The tradeoff in energy consumption between mobility and transmission is exploited by configuring the positions of mobile relays. (2) We study the effect of the initial configuration on the final result. We compare different initial tree building strategies and propose an optimal tree construction strategy for static nodes with no mobility. (3) We develop two algorithms that iteratively refine the configuration of mobile relays. The first improves the tree topology by adding new nodes. It is not guaranteed to find the optimal topology. The second improves the routing tree by relocating nodes without changing the tree topology. It converges to the optimal node positions for the given topology. Our algorithms have efficient distributed implementations that require only limited, localized synchronization. (4) We conduct extensive simulations based on realistic energy models obtained from existing mobile and static sensor platforms. Our results show that our algorithms can reduce energy consumption by up to 45% compared to the best existing solutions.

The rest of the paper is organized as follows. Section 2 reviews related work. In Section 3, we formally define the problem of optimal mobile relay configuration. In Section 4, we present our centralized optimization framework, an optimal solution for the base case with a single mobile relay, an optimal algorithm for constructing a routing tree given no mobility of nodes, and a greedy algorithm for improving the routing tree by adding new nodes. In Section 5, we present an optimal relocation algorithm given a fixed routing topology. In Section 6, we discuss the efficiency and optimality of our framework. In Section 7, we propose efficient distributed implementations for our algorithms. Section 8 describes our simulation results and Section 9 concludes this paper.

2 Related Work

We review three different approaches, mobile base stations, data mules, and mobile relays, that use mobility to reduce energy consumption in wireless sensor networks. A mobile base station moves around the network and collects data from the nodes. In some work, all nodes are always performing multiple hop transmissions to the base station, and the goal is to rotate which nodes are close to the base station in order to balance the transmission load [4], [5], [6]. In other work, nodes only transmit to the base station when it is close to them (or a neighbor). The goal is to compute a mobility path to collect data from visited nodes before those nodes suffer buffer overflows [7], [8], [14], [15]. In [8], [19], [20], several rendezvous-based data collection algorithms are proposed, where the mobile base station only visits a selected set of nodes referred to as rendezvous points within a deadline and the rendezvous points buffer the data from sources. These approaches incur high latencies due to the low to moderate speed, e.g. 0.1-1 m/s [14], [16], of mobile base stations.

Data mules are similar to the second form of mobile base stations [9], [10], [11]. They pick up data from the sensors and transport it to the sink. In [21], the data mule visits all the sources to collect data, transports data over some distance, and then transmits it to the static base station through the network. The goal is to find a movement path that minimizes both communication and mobility energy consumption. Similar to mobile base stations, data mules introduce large delays since sensors have to wait for a mule to pass by before starting their transmission.

In the third approach, the network consists of mobile relay nodes along with static base station and data sources. Relay nodes do not transport data; instead, they move to different locations to decrease the transmission costs. We use the mobile relay approach in this work. Goldenberg et al. [13] showed that an iterative mobility algorithm where each relay node moves to the midpoint of its neighbors converges on the optimal solution for a single routing path. However, they do not account for the cost of moving the relay nodes. In [22], mobile nodes decide to move only when moving is beneficial, but the only position considered is the midpoint of neighbors.

Unlike mobile base stations and data mules, our OMRC problem considers the energy consumption of both mobility and transmission. Our approach also relocates each mobile relay only once immediately after deployment. Unlike previous mobile relay schemes [13] and [22], we consider all possible locations as possible target locations for a mobile node instead of just the midpoint of its neighbors.

Mobility has been extensively studied in sensor network and robotics applications which consider only mobility costs but not communication costs. For

example, in [23], the authors propose approximation algorithms to minimize maximum and total movement of the mobile nodes such that the network becomes connected. In [24], the authors propose an optimal algorithm to bridge the gap between two static nodes by moving nearby mobile nodes along the line connecting the static points while also minimizing the total/maximum distance moved. In [25], [26], the authors propose algorithms to find motion paths for robots to explore the area and perform a certain task while taking into consideration the energy available at each robot. These problems ignore communication costs which add an increased complexity to OMRC, and consequently their results are not applicable.

Our OMRC problem is somewhat similar to a number of graph theory problems such as the Steiner tree problem [27], [28], [29] and the facility location problem [30], [31]. However, because the OMRC cost function is fundamentally different from the cost function for these other problems, existing solutions to these problems cannot be applied directly and do not provide good solutions to OMRC. For example, there is no obvious way to include mobility costs in the Steiner tree problem.

3 Problem Definition

3.1 Energy Consumption Models

Nodes consume energy during communication, computation, and movement, but communication and mobility energy consumption are the major cause of battery drainage. Radios consume considerable energy even in an idle listening state, but the idle listening time of radios can be significantly reduced by a number of sleep scheduling protocols [32]. In this work, we focus on reducing the *total* energy consumption due to transmissions and mobility. Such a holistic objective of energy conservation is motivated by the fact that mobile relays act the same as static forwarding nodes after movement.

For mobility, we consider wheeled sensor nodes with differential drives such as Khepera [17], Robomote [16] and FIRA [18]. This type of node usually has two wheels, each controlled by independent engines. We adopt the distance proportional energy consumption model which is appropriate for this kind of node [33]. The energy $E_M(d)$ consumed by moving a distance d is modeled as:

$$E_M(d) = kd$$

The value of the parameter k depends on the speed of the node. In general, there is an optimal speed at which k is lowest. In [33], the authors discuss in detail the variation of the energy consumption with respect to the speed of the mote. When the node is running at optimal speed, $k = 2$ [33].

To model the energy consumed through transmissions, we analyze the empirical results obtained by two radios CC2420 [34] and CC1000 [35] that are widely used on existing sensor network platforms. For CC2420, the authors of [36] studied the transmission power level needed for transmitting packets reliably (e.g., over 95% packet reception ratio) over different distances. Let $E_T(d)$ be the energy consumed to transmit reliably over distance d. It can be modeled as

$$E_T(d) = m(a + bd^2)$$

where m is the number of bits transmitted and a and b are constants depending on the environment. We now discuss the instantiation of the above model for both CC2420 and CC1000 radio platforms. In an outdoor environment, for received signal strength of -80 dbm (which corresponds to a packet reception ratio higher than 95%), we obtain $a = 0.6 \times 10^{-7} J/bit$ and $b = 4 \times 10^{-10} Jm^{-2}/bit$ from the measurements on CC2420 in [36]. This model is consistent with the theoretical analysis discussed in [37]. We also consider the energy needed by CC1000 to output the same levels. We get lower consumption parameters: $a = 0.3 \times 10^{-7} J/bit$ and $b = 2 \times 10^{-10} Jm^{-2}/bit$. We will see in Section 5 that we maintain this high packet reception ratio throughout our algorithm. We note that although the mobility parameter k is roughly 10^{10} times larger than the transmission parameter b, the relays move only once whereas large amounts of data are transmitted. For large enough data chunk sizes, the savings in energy transmission costs compensates for the energy expended to move the nodes resulting in a decrease in total energy consumed.

3.2 An Illustrative Example

We now describe the main idea of our approach using a simple example. Suppose we have three nodes s_1, s_2, s_3 located at positions x_1, x_2, x_3, respectively (Fig. 1), such that s_2 is a mobile relay node. The objective is to minimize the total energy consumption due to both movement and transmissions. Data storage node s_1 needs to transmit a data chunk to sink s_3 through relay node s_2. One solution is to have s_1 transmit the data from x_1 to node s_2 at position x_2 and node s_2 relays it to sink s_3 at position x_3; that is, node s_2 does not move. Another solution, which takes advantage of s_2's mobility, is to move s_2 to the midpoint of the segment x_1x_3, which is suggested in [13]. This will reduce the transmission energy by reducing the distances separating the nodes. However, moving relay node s_2 also consumes energy. We assume the following parameters for the energy models: $k = 2, a = 0.6 \times 10^{-7}, b = 4 \times 10^{-10}$.

In this example, for a given data chunk m_i, the optimal solution is to move s_2 to x_2^i (a position that we can compute precisely). This will minimize the total energy consumption due to both transmission and mobility. For small messages, s_2 moves very little if at all. As the size of the data increases, relay node

s_2 moves closer to the midpoint. In this example, it is beneficial to move when the message size exceeds 4 MB. We illustrate in Table 1 the energy savings achieved using our optimal approach and the other two approaches for the relevant range of data sizes. For large enough data chunks ($\approx$ 13 MB), one relay node can reduce total energy consumption by 10% compared to the other two approaches. As the data chunk size increases further, the energy savings decrease, and the optimal position converges to the midpoint when the data size exceeds 43 MB. In general, the reduction in energy consumption is higher when there are multiple mobile relay nodes.

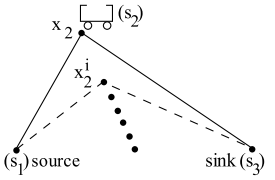


Fig. 1. Reduction in energy consumption due to mobile relay. As the data chunk size increases, the optimal position converges to the midpoint of $s_1 s_3$.

TABLE 1
Energy consumption comparison

Data Size (MB)	Costs at Original Pos.	Costs at Midpoints	Costs at Optimal Pos.	Reduction
5.00	42.78	70.71	42.04	1.73%
11.00	94.12	101.93	88.39	6.09%
12.00	102.68	107.13	94.71	7.75%
13.00	111.23	112.33	100.87	9.32%
14.00	119.79	117.53	106.89	9.06%
15.00	128.35	122.74	112.80	8.09%
16.00	136.90	127.94	118.62	7.28%
17.00	145.46	133.14	124.37	6.58%
18.00	154.01	138.34	130.06	5.98%
40.00	342.26	252.77	247.58	2.05%

The above example illustrates two interesting results. The optimal position of a mobile relay is not the midpoint between the source and sink when both mobility and transmissions costs are taken into consideration. This is in contrast to the conclusion of several previous studies [12], [13] which only account for transmission costs. Second, the optimal position of a mobile relay depends on not only the network topology (e.g., the initial positions of nodes) but also the amount of data to be transmitted. Moreover, as the data chunk size increases, the optimal position converges to the midpoint of s_1 and s_3. These results are particularly important for minimizing the energy cost of data-intensive WSNs as the traffic load of such networks varies significantly with the sampling rates of nodes and network density.

3.3 Problem Formulation

In our definitions, we assume that all movements are completed before any transmissions begin. We also assume there are no obstacles that affect mobility or transmissions. In this case, as we show in Section 4.2, the distance moved by a mobile relay is no more than the distance between its starting position and its corresponding position in the evenly spaced configuration which often leads to a short delay in mobile relay relocation. Furthermore, we assume that all mobile nodes know their locations either by GPS units mounted on them or a localization service in the network. We focus on the case where all nodes are in a 2-dimensional plane $\Re^2$, but the results apply to $\Re^3$ and other metric spaces.

Our problem can be described as follows. Given a network containing one or more static source nodes that store data gathered by other nodes, a number of mobile relay nodes and a static sink, we want to find a directed routing tree from the sources to the sink as well as the optimal positions of the mobile nodes in the tree in order to minimize the total energy consumed by transmitting data from the source(s) to the sink and the energy consumed by relocating the mobile relays. The source nodes in our problem formulation serve as *storage points* which cache the data gathered by other nodes and periodically transmit to the sink, in response to user queries. Such a network architecture is consistent with the design of storage-centric sensor networks [38]. Our problem formulation also considers the initial positions of nodes and the amount of data that needs to be transmitted from each storage node to the sink. The formal definition of the problem is given below.

Definition 1: (Optimal Mobile Relay Configuration): Input Instance: S, a list of n nodes $(s_1, \ldots, s_n)$ in the network; O, a list of n locations $(o_1, \ldots, o_n)$ where o_i is the initial position of node s_i for $1 \leq i \leq n$; $S_{sources}$, a subset of S representing the source nodes; r, a node in S, representing the single sink; $M_{sources} = \{M_i \mid s_i \in S_{sources}\}$, a set of data chunk sizes for all sources in $S_{sources}$;

We define m_i, which we compute later, to be the weight of node s_i which is equal to the total number of bits to be transmitted by node s_i. We define a configuration $\langle E, U \rangle$ as a pair of two sets: E, a set of directed arcs (s_i, s_j) that represent the directed tree in which all sources are leaves and the sink is the root and U, a list of locations $(u_1, \ldots, u_n)$ where u_i is the transmission position for node s_i for $1 \leq i \leq n$. The cost of a configuration $\langle E, U \rangle$ is given by:

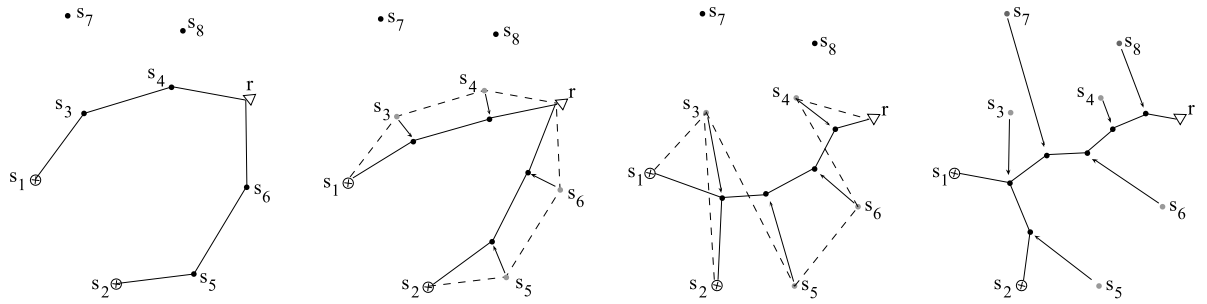
$$c(\langle E, U \rangle) = \sum_{(s_i, s_j) \in E} a m_i + b \|u_i - u_j\|^2 m_i + k \|o_i - u_i\|$$

Output: $\langle E, U \rangle$, an optimal configuration that minimizes the cost $c(\langle E, U \rangle)$.

4 Centralized Solution

4.1 Energy Optimization Framework

The Optimal Mobile Relay Configuration (OMRC) problem is challenging because of the dependence of



(a) Optimal routing tree for $0 \leq m \leq 15$MB.

(b) Optimal configuration for 15MB$\leq m \leq$ 25MB: same topology but nodes relocate.

(c) Optimal configuration for 25MB$\leq m \leq 60$ MB. A new topology is used.

(d) Optimal configuration for 100MB$\leq m \leq$ 150MB: a new topology with more nodes.

Fig. 2. Example of optimal configurations as a function of amount of data to be transferred. In each part, source nodes s_1 and s_2 must send m bits of data to the sink r. We consider m up to 150MB.

the solution on multiple factors such as the routing tree topology and the amount of data transferred through each link. For example, when transferring little data, the optimal configuration is to use only some relay nodes at their original positions. As the amount of data transferred increases, three changes occur: the topology may change by adding new relay nodes, the topology may change by changing which edges are used, and the relay nodes may move closer together. In many cases, we may have restrictions such as no mobility for certain relay nodes or we must use a fixed routing tree. These constraints affect the optimal configuration.

We illustrate how the optimal configuration depends on the amount of data to transfer using the example from Fig. 2a. When there is very little data to transfer, the optimal routing tree T_a depicted in Fig. 2a uses only some of the relay nodes in their original positions. When the amount of data to transfer from s_1 and s_2 increases to 15 MB, the relay nodes in tree T_a move to their corresponding positions in tree T_b of Fig. 2b but the topology does not change. When the amount of data to transfer from s_1 and s_2 is between 25 and 60 MB, the optimal routing tree has a different topology as shown in Fig. 2c. For even larger messages, new trees with even more nodes included are optimal. For example, when the amount of data to be transferred is between 100 and 150 MB, the optimal tree is depicted in Fig. 2d. No existing tree construction strategy handles all these cases. For example, the minimum spanning tree that includes all network nodes has two fundamental problems. It will typically include unneeded nodes, and it typically creates non-optimal topologies as it focuses only on the current location of nodes as opposed to where nodes may move to.

We now present a centralized approach to solve OMRC that breaks the problem into three distinct steps: initial tree construction, node insertions, and tree optimization. For each step, we present an algorithm to solve the corresponding subproblem. Our algorithm for initial tree construction is optimal for the static environment where nodes cannot move. However, we can effectively apply the later algorithms if we must start with a different topology. Our greedy heuristic for improving the routing tree topology by adding nodes exploits the mobility of the newly added nodes. Our tree optimization algorithm improves the routing tree by relocating its nodes without changing its topology. This iterative algorithm converges on the optimal position for each node given the constraint that the routing tree topology is fixed. Our node insertion and tree optimization algorithms use the $LocalPos$ algorithm we propose in Fig. 3 that optimally solves the simplest case (see Section 4.2) of the mobile relay configuration problem where there is a single source, a single sink, and a single relay node. Our approach is not guaranteed to produce an optimal configuration because we do not necessarily find the optimal topology, but our simulation results show that it performs well.

4.2 Base Case

Before presenting our algorithm for OMRC, we revisit the example of Section 3.2 as it represents the simplest possible base case of the problem in which the network consists of one source s_{i-1}, one mobile relay node s_i and one sink s_{i+1}. In this section, we calculate the optimal position for the relay node. We use the following notation. In $\Re^2$, let the original position of a node s_j be $o_j = (p_j, q_j)$, and let $u_j = (x_j, y_j)$ its final position in configuration U. According to our energy models, the total transmission and movement energy cost incurred by the mobile relay node s_i is

$$c_i(U) = k\|u_i - o_i\| + am + b\|u_{i+1} - u_i\|^2 m$$

We also define

$$C_i(U) = c_i(U) + am + b\|u_i - u_{i-1}\|^2 m$$

This corresponds to the transmission cost of node s_{i-1} plus the total cost of node s_i, which is the total cost of the final configuration in this example. We need to compute a position u_i for s_i that minimizes $C_i(U)$ assuming that $u_{i-1} = o_{i-1}$ and $u_{i+1} = o_{i+1}$; that is, node s_i's neighbors remain at the same positions in the final configuration U. We calculate position $u_i = (x_i, y_i)$ for node s_i by finding the values for x_i and y_i where the partial derivatives of the cost function $C_i(U)$ with respect to x_i and y_i become zero. Position u_i will be toward the midpoint of positions u_{i-1} and u_{i+1}. The partial derivatives $\frac{\delta C_i(U)}{\delta x_i}$, $\frac{\delta C_i(U)}{\delta y_i}$ at x_i and y_i, respectively are defined as follows.

$$\frac{\delta C_i(U)}{\delta x_i} = -2bm(x_{i+1} - x_i) + 2bm(x_i - x_{i-1}) + k\frac{(x_i - p_i)}{\sqrt{(x_i - p_i)^2 + (y_i - q_i)^2}}$$

$$\frac{\delta C_i(U)}{\delta y_i} = -2bm(y_{i+1} - y_i) + 2bm(y_i - y_{i-1}) + k\frac{(y_i - q_i)}{\sqrt{(x_i - p_i)^2 + (y_i - q_i)^2}}$$

Setting $\frac{\delta C_i(U)}{\delta x_i} = 0, \frac{\delta C_i(U)}{\delta y_i} = 0$, we get the following two cases. Suppose s_i needs to move left. This means p_i is to the right of the midpoint of nodes s_{i-1} and s_{i+1}. Let $Y_i = \frac{k}{4bm}\frac{1}{\sqrt{1+\frac{(y_{i-1}+y_{i+1}-2q_i)^2}{(x_{i-1}+x_{i+1}-2p_i)^2}}}$. The optimal position is then $x_i = \frac{1}{2}(x_{i-1} + x_{i+1}) + Y_i$. If s_i needs to move right, then p_i is to the left of the midpoint of nodes s_{i-1} and s_{i+1}. The optimal position is then $x_i = \frac{1}{2}(x_{i-1} + x_{i+1}) - Y_i$. The corresponding y_i in both cases is $\frac{(x_{i-1}+x_{i+1}-2p_i)}{(y_{i-1}+y_{i+1}-2q_i)}(x_i - p_i) + q_i$.

We note that in some cases it might not be beneficial to move, so the optimal position for the relay node is its original position. The algorithm to compute the optimal position of a relay node given its neighbors is shown in Fig. 3.

```
function LOCALPOS(o_i, u_i, u_{i-1}, u_{i+1})
    ▷ Consider case s_i moves right
    valid ← FALSE;
    x_i ← ½(x_{i-1} + x_{i+1}) − Y_i;
    if x_i > p_i then
        valid ← TRUE;
    else
        ▷ Consider case s_i moves left
        x_i ← ½(x_{i-1} + x_{i+1}) + Y_i
        if x_i < p_i then
            valid ← TRUE;
        end if
    end if
    ▷ Record if new position is different from previous one
    if valid then
        y_i ← ((x_{i-1}+x_{i+1}−2p_i)/(y_{i-1}+y_{i+1}−2q_i))(x_i − p_i) + q_i;
        u'_i = (x_i, y_i);
        if ||u'_i − u_i|| > threshold then
            return (u'_i, TRUE);
        end if
    end if
    ▷ not beneficial to move, stay at original position
    return (o_i, FALSE);
end function
```

Fig. 3. Algorithm to compute the optimal position of a relay node that receives data from a single node and transmits the data to a single node.

4.3 Static Tree Construction

Different applications may apply different constraints on the routing tree. When only optimizing energy consumption, a shortest path strategy (as discussed below) yields an optimal routing tree given no mobility of nodes. However, in some applications, we do not have the freedom of selecting the routes. Instead, they are predetermined according to some other factors (such as delay, capacity, etc). In other less stringent cases, we may be able to update the given routes provided we keep the main structure of the tree. Depending on the route constraints dictated by the application, we start our solution at different phases of the algorithm. In the unrestricted case, we start at the first step of constructing the tree. When the given tree must be loosely preserved, we start with the relay insertion step. Finally, with fixed routes, we apply directly our tree optimization algorithm. Our simulations (Section 8) show that our approach outperforms existing approaches for all these cases.

We construct the tree for our starting configuration using a shortest path strategy. We first define a weight function w specific to our communication energy model. For each pair of nodes s_i and s_j in the network, we define the weight of edge s_is_j as: $w(s_i, s_j) = a + b\|o_i - o_j\|^2$ where o_i and o_j are the original positions of nodes s_i and s_j and a and b are the energy parameters discussed in Section 3.1. We observe that using this weight function, the optimal tree in a static environment coincides with the shortest path tree rooted at the sink. So we apply Dijkstra's shortest path algorithm starting at the sink to all the source nodes to obtain our initial topology.

4.4 Node Insertion

We improve the routing tree by greedily adding nodes to the routing tree exploiting the mobility of the inserted nodes. For each node s_{out} that is not in the tree and each tree edge s_is_j, we compute the reduction (or increase) in the total cost along with the optimal position of s_{out} if s_{out} joins the tree such that data is routed from s_i to s_{out} to s_j instead of directly from s_i to s_j using the LocalPos algorithm described in Fig. 3. We repeatedly insert the outside node with the highest reduction value modifying the topology to include the selected node at its optimal position, though the node will not actually move until the completion of the tree optimization phase. After each node insertion occurs, we compute the reduction in total cost and optimal

position for each remaining outside node for the two newly added edges (and remove this information for the edge that no longer exists in the tree). At the end of this step, the topology of the routing tree is fixed and its mobile nodes can start the tree optimization phase to relocate to their optimal positions.

5 Tree Optimization

In this section, we consider the subproblem of finding the optimal positions of relay nodes for a routing tree given that the topology is fixed. We assume the topology is a directed tree in which the leaves are sources and the root is the sink. We also assume that separate messages cannot be compressed or merged; that is, if two distinct messages of lengths m_1 and m_2 use the same link (s_i, s_j) on the path from a source to a sink, the total number of bits that must traverse link (s_i, s_j) is $m_1 + m_2$.

First, we extend the base case solution of Section 4.2 to handle multiple flows passing through a mobile relay node. Then, we propose an iterative algorithm that uses the solution for this base case to compute the new positions of the relay nodes in the routing tree. We also show that this algorithm converges to the optimal solution for the given tree given the topology is fixed.

5.1 Extended Base Case

Before we describe our optimal algorithm for this problem, we extend the solution to the base case presented in Section 4.2 to the more general multiple flow traffic pattern. The network now consists of multiple sources, one relay node and one sink such that data is transmitted from each source to the relay node and then to the sink. We modify our solution as follows. Let s_i be the mobile relay node, $S(s_i)$ the set of source nodes transmitting to s_i and s_i^d the sink collecting nodes from s_i. The cost incurred by s_i in this configuration U is:

$$c_i(U) = k\|u_i - o_i\| + am_i + bm_i\|u_d - u_i\|^2$$

where m_i is the total amount of data that s_i transmits to s_i^d. Similar to the single source base case, we define

$$C_i(U) = c_i(U) + \sum_{s_l \in S(s_i)} am_l + b\|u_i - u_l\|^2 m_l$$

This corresponds to the transmission cost of all nodes s_l that send messages to node s_i plus the total cost of node s_i. In this case, this also corresponds to the total cost of configuration U which we wish to minimize. First, we compute m_i as $\sum_{s_l \in S(s_i)} m_l$. We then follow the same routine of computing the points at which both partial derivatives $\frac{\delta C_i(U)}{\delta x_i}$ and $\frac{\delta C_i(U)}{\delta y_i}$ become zero. We obtain the following positions:

$$x_i = p_i + \frac{-B_x(\sqrt{B_x^2 + B_y^2} \pm k)}{A\sqrt{B_x^2 + B_y^2}} \quad y_i = q_i + \frac{-B_y(\sqrt{B_x^2 + B_y^2} \pm k)}{A\sqrt{B_x^2 + B_y^2}}$$

```
procedure OPTIMALPOSITIONS(U^0)
    converged ← false;
    j ← 0;
    repeat
        anymove ← false;
        j ← j + 1;
        ▷ Start an even iteration followed by an odd iteration
        for idx = 2 to 3 do
            for i = idx to n by 2 do
                (u_i^j, moved) ← LOCALPOS(o_i, S(s_i), s_i^d);
                anymove ← anymove OR moved
            end for
        end for
        converged ← NOT anymove
    until converged
end procedure
```

Fig. 4. Centralized algorithm to compute the optimal positions in a given tree

where

$$A = m_i + \sum_{s_l \in S(s_i)} m_l$$

$$B_x = m_i x_d + \sum_{s_l \in S(s_i)} m_l x_l + Ap_i$$

$$B_y = m_i y_d + \sum_{s_l \in S(s_i)} m_l y_l + Aq_i$$

We note that these values correspond to two candidate points moving in each direction (left/right). The optimal position is the valid value yielding the minimum cost.

5.2 Optimization Algorithm

We propose a simple iterative approach to compute the optimal position u_i for each node s_i. We define the following notations. Let $u_i^j = (x_i^j, y_i^j)$ be the position of node s_i after the jth iteration of our algorithm for $j \geq 0$ and $U^j = (u_1^j, \ldots, u_n^j)$ the computed configuration of nodes s_1 through s_n after j iterations. We define $u_i^0 = o_i$. Note that the mobile relay nodes do not move until the final positions are computed.

Our algorithm starts by an odd/even labeling step followed by a weighting step. To obtain consistent labels for nodes, we start the labeling process from the root using a breadth first traversal of the tree. The root gets labeled as even. Each of its children gets labeled as odd. Each subsequent child is then given the opposite label of its parent. We define m_i, the weight of a node s_i, to be the sum of message lengths over all paths passing through s_i. This computation starts from the sources or leaves of our routing tree. Initially, we know $m_i = M_i$ for each source leaf node s_i. For each intermediate node s_i, we compute its weight as the sum of the weights of its children.

Once each node gets a weight and a label, we start our iterative scheme. In odd iterations j, the algorithm computes a position u_i^j for each odd-labeled node s_i that minimizes $C_i(U^j)$ assuming that $u_{i-1}^j = u_{i-1}^{j-1}$ and

$u_{i+1}^j = u_{i+1}^{j-1}$; that is, node s_i's even numbered neighboring nodes remain in place in configuration U^j. In even-numbered iterations, the controller does the same for even-labeled nodes. The algorithm behaves this way because the optimization of u_i^j requires a fixed location for the child nodes and the parent of s_i. By alternating between optimizing for odd and even labeled nodes, the algorithm guarantees that the node s_i is always making progress towards the optimal position u_i. Our iterative algorithm is shown in Fig. 4

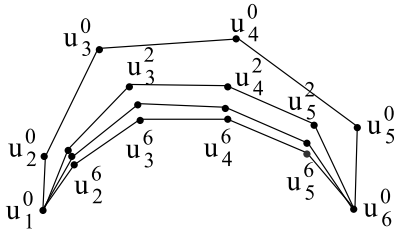


Fig. 5. Convergence of iterative approach to the optimal solution. Each line shows the configuration obtained after 2 iterations. The optimal configuration is reached after 6 iterations.

Fig. 5 shows an example of an optimal configuration for a simple tree with one source node. Nodes start at configuration U^0. In the first iteration, odd nodes (s_3 and s_5) moved to their new positions (u_3^1, u_5^1) computed based on the current location of their (even) neighbors (u_2^0, u_4^0, u_6^0). In the second iteration, only even nodes (s_2 and s_4) moved to their new positions (u_2^2, u_4^2) computed based on the current location of their (odd) neighbors (u_1^1, u_3^1, u_5^1). Since s_3 and s_5 did not move, their position at the end of this iteration remains the same, so $u_3^1 = u_3^2$ and $u_5^1 = u_5^2$. In this example, nodes did two more sets of iterations, and finally converged to the optimal solution shown by configuration U^6.

Even though configurations change with every iteration, nodes only move after the final positions have been computed. So each node follows a straight line to its final destination. As the data size increases, nodes in the optimal configuration get more evenly spaced. In fact, in any given configuration, the maximum distance traveled by a node is bounded by the distance between its starting position and its final position in the evenly spaced configuration.

The above example shows another property of our algorithm. When a node s_i moves and its neighbors (s_{i-1} and s_{i+1}) remain in place, it moves in the direction of the midpoint of $s_{i-1}s_{i+1}$. This results in a reduction in the length of one of the transmission links. The other may increase in length but will never exceed the new length of the first link. This remains valid for multiple children case. So in any configuration U^{i+1}, the length of the largest link is at most the length of the largest link in the previous configuration U^i. So if we start with a route along good quality links, this quality will be preserved in the optimal configuration (and throughout intermediate configurations).

6 Efficiency and Optimality

We first consider efficiency. Our initial tree construction algorithm is essentially a single source shortest path algorithm. Using Dijkstra's algorithm, the time complexity is $O(n^2)$ where n is the number of nodes. Our second algorithm needs to compute the reduction in cost for each pair of node and tree edge, so the time complexity is $O(n^2)$. Our tree optimization algorithm runs until the change in position for each node falls below a predefined threshold. The value of this threshold represents a tradeoff between precision and cost. As the threshold decreases, more iterations are needed for convergence. Upon termination, no node can move by itself to improve the overall cost (within the threshold bound). We have not completed a rate of convergence analysis for this algorithm. However, in our simulations, we reach our error threshold within 8 to 10 iterations. Since each iteration involves only half the nodes and each computation of u_i^j can be performed in constant time, the time complexity of our algorithm is $O(\lambda n)$, where λ is the number of iterations to reach convergence. Given that $\lambda \leq 10$ in our simulations, our observed time complexity is $O(n)$. The resulting time complexity for the full approach is $O(n^2)$.

With respect to optimality, our resulting configuration is not necessarily optimal because we do not necessarily find the optimal topology. However, two of our algorithms, the initial tree construction algorithm and the tree optimization algorithm, are optimal for their respective subproblems. That is, our initial tree construction algorithm is optimal in a static environment where nodes cannot move so that only the original positions of the nodes are considered. Likewise, for our tree optimization algorithm, we prove that the final configuration where no node can move by itself to improve the overall cost (within the threshold bound) is globally optimal; that is, no simultaneous relocation of multiple nodes can improve the overall cost. We present the proof of optimality in the appendix.

7 Distributed Algorithms

Our solutions to the three subproblems assume a centralized scheme in which one node has full knowledge of the network including which nodes are on the transmission paths to each source, the original physical position o_i of each node s_i, and the total message length m to be sent from each source. Whereas the centralized algorithm computes the optimal static tree and the optimal position of each node in the restructured tree, it incurs prohibitively high overhead in large-scale networks. We now present a distributed and decentralized version of each of our algorithms.

We modify the first phase, the tree construction phase, to use a fully distributed routing algorithm. We pick greedy geographic routing since it does not require global knowledge of the network although any algorithm with such property can be used.

After a routing tree is constructed, the tree restructuring phase begins. Network nodes outside the tree broadcast their availability (as NODE_IN_RANGE message) to tree nodes within their communication range and wait for responses for a period of time T_w. Similarly, tree nodes enter a listening phase T_u. During that period, tree nodes receive messages of different types (NODE_IN_RANGE, OFFER, ...). Each tree node that receives one or more NODE_IN_RANGE message responds to the sender by giving it its location information and its parent's location information. Each non-tree node s_o that receives location information from a tree node s_i during T_w computes the reduction in cost if it joins the tree as parent of s_i and adds s_i to a list of candidates. At the end of T_w, the non-tree node selects from the candidate list the node that results in the largest reduction and sends it an offer. It also sends the tree node with the second largest reduction a POTENTIAL_OFFER message. At the end of T_u, each tree node v_t that collected one or more offers and potential offers operates as follows. If v_t's best potential offer exceeds its best offer by a certain threshold B and v_t has not already waited R rounds, v_t waits rather than accepting its best offer in the hopes that its best potential offer will become an actual offer in another round. By waiting, it sends everyone a REJECT_OFFER, restarts the listening phase, and records that it has waited another round. Otherwise, v_t accepts its best offer by responding to its sender p with an ACCEPT_OFFER message and to the remaining nodes with a REJECT_OFFER message. It then updates its parent in the tree to p, resets T_u and starts the listening phase again.

A non-tree node p that receives an ACCEPT_OFFER message moves to the corresponding local optimal location and joins the tree. It becomes a tree node and enters the listening phase. On the other hand, if p does not receive an ACCEPT_OFFER, p repeats the process by broadcasting its availability again and resetting T_w. We note that values in p's candidate list cannot be reused to extend offers to old tree nodes since those tree nodes could have a new parent at this point in time. When the second phase ends, any remaining non-tree nodes stop processing whereas tree nodes enter the tree optimization phase. Fig. 6 shows the algorithm executed by each tree node.

Giving tree nodes the ability to wait before accepting an offer increases the chances of using mobile relay nodes to their full potential. For example, consider a scenario where several mobile relay nodes can greatly improve the capacities of several tree links but are all closest to one specific link. They will all send offers to the same tree node while the rest of the tree nodes in their proximity will receive modest offers from more distant mobile nodes. If the tree nodes cannot wait, they will be forced to accept a modest offer and the mobile nodes will either remain unused or they will help more distant tree nodes where their impact is reduced since they use up more energy to get to their new location.

```
procedure TREERUN
  ▷ Phase I: Run routing algorithm to discover parent and children
  (parent, children) ← DISTRIBUTEDROUTING;
  ▷ Phase II: Start tree restructuring phase
  offers ← ∅; potentialoffers ← ∅; wait ← 0;
  repeat
    ▷ Listen to incoming offers or changes in structure
    repeat
      RECEIVE(sender, type, data);
      if type = MOBILE_IN_RANGE then
        SEND(sender, META_DATA, info);
      else if type = OFFER then
        offers.add(data);
      else if type = POTENTIAL_OFFER then
        potentialloffers.add(data);
      else if type = UPDATE_STRUCTURE then
        children.add(data.newchild);
        children.remove(data.oldchild);
      end if
    until timeout
    ▷ Process offers and pick best
    if offers ≠ ∅ then
      bestOffer ← offers.dequeue();
      bestPotentialOffer ← potentialoffers.dequeue();
      if bestPotentialOffer > bestOffer*B and wait < R then
        SEND(bestOffer.sender, REJECT_OFFER);
        wait++;
      else
        SEND(bestOffer.sender, ACCEPT_OFFER);
        parent ← bestOffer.sender;
      end if
    end if
    while offers ≠ ∅ do
      offer ← candidates.dequeue();
      SEND(offer.sender, REJECT_OFFER);
    end while
  until timeout
  ▷ Phase III: Iterate moving to optimal local positions
  converged ← false;
  while not converged do
    (u, converged) ← LOCALPOS(o, parent, children);
    ▷ Exchange location info with parent and children
    SEND(parent, NEW_LOCATION, u);
    for all child ∈ children do
      SEND(child, NEW_LOCATION, u);
    end for
    RECEIVE(parent, NEW_LOCATION, parent.u);
    for all child ∈ children do
      RECEIVE(child, NEW_LOCATION, child.u);
    end for
  end while
end procedure
```

Fig. 6. Local algorithm executed by tree nodes

The centralized tree optimization algorithm can be transformed into a distributed algorithm in a natural way. The key observation is that computing each u_i^j for node s_i only depends on the current position of s_i's neighbors in the tree (children and parent), nodes that s_i normally communicates with for data transfers. Thus, s_i can perform this computation. The distributed implementation proceeds as follows. First,

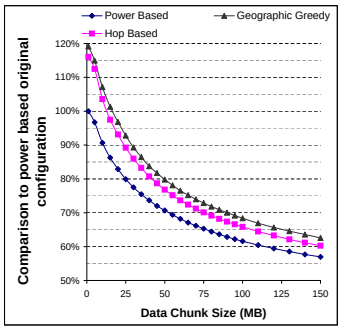


Fig. 7. Graph of the average static energy consumption ratio of TREE+INS+FO as a function of data chunk size for our three tree construction strategies PB, HB, and GG

there is a setup process where the sender s_1 sends a discover message that ends with the receiver s_n; the two purposes of this message are (1) to assign a label of odd or even to each node s_i and (2) for each node s_i to learn the current positions of its neighbors. A node s_i sends its current position to node s_j when acknowledging receipt of the discover message. Second, there is a distributed process by which the nodes compute their transmission positions. We make each iteration of the basic algorithm a "round", though there does not need to be explicit synchronization. In odd rounds, each odd node computes its locally optimal position and transmits this new position to its neighbors. In even rounds, each even node does the same. A node begins its next round when it receives updated positions from all its neighbors. The final step is to have the nodes move to their computed transmission positions, send messages to their neighbors saying they are in position, and finally perform the transmission. To ensure the second process does not take too long, we limit the number of rounds to 8; that is, each node computes an updated position four times. Simulation results show that this is enough to obtain costs close to optimal (see Section 8).

8 Simulations

We carried out simulations on 100 randomly generated initial topologies, each of which has 100 nodes placed uniformly at random within a 150m by 150m area. We used these initial topologies to generate two subsequent sets of complete topologies with established sources and sink. We used the first set to study the effectiveness of our algorithms as the amount of data transferred to the sink varies and the second set to study the effectiveness of our algorithms for different numbers of sources. In the first set, we selected sources and sinks uniformly at random from these 100 nodes. We varied the number of sources from 4 to 12, by increments of 2, and used each number of sources for 20 initial topologies. For each resulting topology, we created many separate input instances by varying the data chunk size from 1MB to 150MB where the data chunk size for an input instance is the common amount of data to be transferred from each source to the sink. In the second set, for each initial topology, we generated 10 different complete topologies by starting with 2 randomly selected sources, and adding two new sources to the previous set at each step.

We used the following settings to model the transmission and mobility costs of our nodes. For transmission, we use $a = 0.6 \times 10^{-7}$ and $b = 4 \times 10^{-10}$ as the standard setting which is consistent with the empirical measurements on CC2420 motes [36]. For mobility, we used different settings in each of our two sets. In the first set, we used $k = 2$ as the standard setting because it models several platforms such as Robomote [16], [17]. In the second set, we set k to be 1, 2 and 4 since we additionally use that set to study the effect of different mobility costs on the energy reduction. Furthermore, we set the maximum communication distance of a node to be 30m, which was shown to result in a high packet reception ratio for the CC2420 radio [36]. We ran simulations using different values for the convergence threshold. We obtained similar gains for values less than or equal to 0.01. In the following simulations, we set the threshold to 0.01.

Our algorithmic framework starts with an initial routing tree. In the centralized setting, we construct this initial routing tree using the following three widely used routing algorithms: power based routing, hop based routing, and greedy geographic routing. Power based routing computes a shortest path from the sink to each source with each edge weight being the square of the distance between the two corresponding nodes plus some constant value to represent the energy consumed $a + bd^2$ to transmit each byte of data over that edge. Hop based routing minimizes the number of hops between each source and the sink and is the base of several widely used algorithms in wireless networks (e.g. AODV [39]). Given our maximum communication range of 30m, we do not have any links with poor quality which is a common concern with hop based routing. Greedy geographic routing is a greedy strategy in which each node forwards messages to the reachable node (within the communication range of the node) that is closest to the sink. The first two tree construction approaches require global knowledge of the network whereas the last one is fully localized. For the distributed setting, we construct the initial routing tree using greedy geographic routing because it is fully localized. Of the 100 initial topologies, the distributed routing algorithm resulted in a disconnected path between the sources and the sink in only four networks given our maximum communication distance of 30m.

We study variants of our strategy where we use only one optimization, inserting nodes or optimizing a given tree, to determine the benefit of both optimizations. Specifically, we use TREE to represent

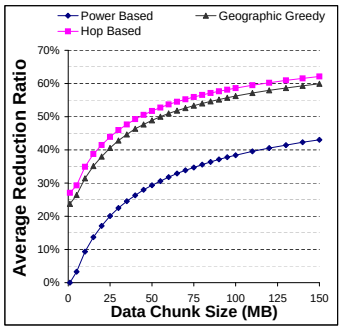


Fig. 8. Graph of the average reduction ratio of optimization INS+FO as a function of data chunk size for our three tree construction strategies PB, HB, and GG

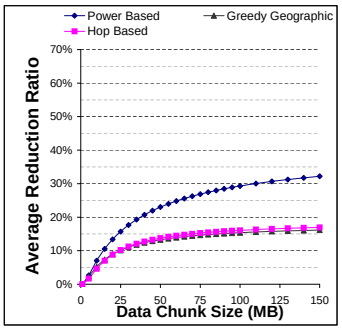


Fig. 9. Graph of the average reduction ratio of optimization FO as a function of data chunk size for our three tree construction strategies PB, HB, and GG

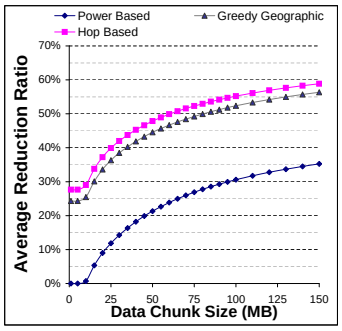


Fig. 10. Graph of the average reduction ratio of optimization INS as a function of data chunk size for our three tree construction strategies PB, HB, and GG

the variant where we only construct an initial tree and do no optimizations, TREE+FO to represent the variant where we optimize the initial tree, TREE+INS to represent the variant where we insert nodes into the initial tree, and TREE+INS+FO to represent the variant where we insert nodes into the initial tree and then optimize the final tree. The three possibilities for TREE are PB, HB, and GG which represent the Power Based, Hop Based, and Greedy Geographic tree construction algorithms, respectively. For each input instance I, we let $TREE(I)$ denote the energy consumed by the initial tree constructed by our three tree construction algorithms PB, HB, and GG, and we let $TREE+OPT(I)$ denote the energy consumed by the final optimized tree where TREE can be PB, HB, or GG and OPT can be INS, FO or INS+FO. The *reduction ratio* achieved by optimization OPT on input I for tree construction algorithm TREE is $(TREE(I)-TREE+OPT(I))/TREE(I)$. We measure the performance of optimization OPT on initial tree strategy TREE by computing the average reduction ratio achieved by OPT over all input instances I of set 1 that have the same data chunk size. Moreover, for each input instance I and each algorithm TREE+OPT, we define the *static energy ratio* $(TREE+OPT(I))/PB(I)$ where $PB(I)$ is the cost of the power based tree which is the optimal cost for the static version of this problem where no nodes can move. The static energy ratio measures the benefit of our algorithms which exploit mobility of nodes versus the static optimal configuration. We measure the overall performance of algorithm TREE+OPT by computing the average static energy ratio achieved by TREE+OPT over all input instances I of set 1 that have the same data chunk size. Finally, we measure the performance of optimization INS+FO on initial tree strategy TREE by computing the average reduction ratio achieved by INS+FO over all input instances I of set 2 that have the same number of sources.

8.1 Centralized Algorithm

We first show the benefit of exploiting the mobility of relay nodes by computing the average static energy consumption ratio of TREE+INS+FO for all data chunk sizes for each of our three tree building strategies PB, HB, and GG as shown in Fig. 7. For all three initial tree strategies, we see that the average static energy consumption ratio drops quickly as the data chunk size increases. For HB and GG, the average static energy consumption ratio starts out higher than 100% because $PB(I)$, the optimal tree for the static case, is roughly 37% lower than $HB(I)$ and $GG(I)$ for any of our input instances. Even given this initial disadvantage of a poor starting tree from an energy consumption perspective, we see that the average static energy consumption ratios of HB+INS+FO and GG+INS+FO drop below 100% for data chunk sizes of 12 MB and 15 MB, respectively. As the data chunk size increases further, both HB+INS+FO and GG+INS+FO achieve average static energy consumption ratios of 75% and 60% for data chunk sizes of 60 MB and 150MB, respectively. The results for PB+INS+FO are even better because we start with the optimal tree for the static case. Thus, the average static energy consumption ratio for PB+INS+FO is always below 100% and reaches 55% for 150 MB.

We now evaluate the benefit achieved by our optimizations FO, INS, and INS+FO for each of our tree building strategies PB, HB, and GG. We note that in this set of simulations, we used our centralized improvement schemes with the distributed tree building approach GG. The purpose is to test the limits of our optimizations given a non-optimal starting tree. A fully distributed setup is studied later in this section.

We start with optimization INS+FO. Fig. 8 plots the average reduction ratio for optimization INS+FO for PB, HB, and GG. In all three cases, we see the same basic trend; the average reduction ratio increases as data chunk size increases. For both HB and GG, the average reduction ratio starts at roughly 25% for

small data chunk sizes and exceeds 60% for large data chunk sizes; for PB the average reduction ratio starts near 0% and exceeds 43% for large data chunk sizes. The difference in average reduction ratio, in particular for small data chunk sizes, is due to the quality of the initial tree. For PB, the initial tree is good so there is little that our optimization INS+FO can do to improve energy consumption for small data chunk sizes. For HB and GG, the initial tree can be very poor, so INS+FO can provide immediate improvement to the tree to significantly reduce energy consumption by an average of 25% for data chunk sizes of only 1MB. We note that although INS+FO achieves higher reduction ratios for HB and GG than for PB, the total energy consumed by PB+INS+FO is lower than the total energy consumed by HB+INS+FO or GG+INS+FO.

We next consider optimization FO alone. Fig. 9 plots the average reduction ratio for optimization FO for PB, HB, and GG. In all three cases, the average reduction ratio starts at 0% for small data chunk sizes and increases to roughly 18% for HB and GG and 33% for PB for large data chunk sizes. It is interesting to note that FO is most effective for PB whereas INS+FO achieved significantly greater reduction ratios for GG and HB for all data chunk sizes.

Finally, we consider optimization INS alone. Fig. 10 plots the average reduction ratio for optimization INS for PB, HB, and GG. In all three cases, we see the average reduction ratio of INS alone is comparable to that of INS+FO (within 5%-8% for data chunk sizes of at least 15MB). For very small data chunk sizes, the average reduction ratio is constant until a certain threshold is exceeded and then rises significantly.

We now evaluate our approach as we vary the number of sources. We used the greedy geographic tree GG as our initial tree and INS+FO as our optimization algorithm. Fig. 11 shows the average reduction ratio as a function of the number of sources. We observe that this ratio remains almost constant for different values of k as the difference in ratios for different number of sources does not exceed 3.5%. Fig. 11 also shows the effect of mobility costs on the reduction in energy consumption costs in general. As mobility costs decrease, it becomes more effective for mobile nodes to move over longer distances and reduce the communication consumption further so the reduction in total costs increases as k decreases.

Given our simulation results, we draw the following five conclusions. First, we achieve the best results when we use the power based tree PB as our initial tree. Second, if we use the power based tree, either optimization alone is very effective and both optimizations together achieve the best results. Third, if we start with either the hop based tree HB or the greedy geographic tree GG, the most effective optimization is the node insertion optimization INS which achieves nearly as good an average reduction ratio as INS+FO. Fourth, if we start with the hop based or greedy geographic tree, we can achieve a static energy ratio that is close to that achieved by starting with the power based tree if we apply both optimizations. In particular, the node insertion optimization INS helps alleviate the initial disadvantage by adding a lot of new nodes into the tree. We briefly explain the reason for all of these conclusions. The key observation is that the hop based and greedy geographic trees HB and GG tend to create initial trees with relatively long edges and relatively few nodes whereas the power based tree PB tends to create trees with lots of nodes and relatively short edges because of the quadratic cost metric. As a result, for HB and GG, optimization FO alone which rearranges nodes is relatively ineffective as it can only balance the relatively long edges. On the other hand, optimization INS alone can insert new nodes into the tree and thus create a new tree with significantly shorter edges on average given HB or GG as the initial tree. Because PB starts with many more nodes and shorter edges, PB does not benefit as much from node insertion INS as HB and GG do, and PB benefits a lot more from node rearrangement FO than HB and GG do. Fifth, the improvement ratios that we obtain are almost independent of the number of sources in the network.

In all our simulation results, the standard deviation varied between 4% and 6.5%. We identified six outlier topologies which deviated from the mean by more than 10%. In these topologies, the sources were either very close to the sink so there was little room for improvement or very far from the sink so the improvement was much greater than the average case.

8.2 Distributed Algorithm

We now evaluate how well our distributed implementation works. Our initial tree is the greedy geographic tree GG. We consider four optimizations: the centralized implementation of INS+FO, the distributed implementation of just FO, the distributed implementation of just INS, and the distributed implementation of INS followed by the distributed implementation of FO. For the distributed implementation of INS, we set parameter B to 10% (a potential offer must be 10% better than the best actual offer to cause a node to wait). Fig. 12 shows the average reduction ratio of each of these optimizations. The average reduction ratio for distributed INS+FO starts at 20% for small data chunk sizes, reaches 30% for data chunk sizes around 20MB, and exceeds 40% for data chunk sizes larger than 75MB. The gap between the average reduction ratio for centralized INS+FO and distributed INS+FO starts at roughly 5% for small data chunk sizes and increases to roughly 15% for large data chunk sizes. This gap is due to the lack of global information when performing the insertion step. Expensive links in the tree that do not have nearby relay nodes are not able to communicate with

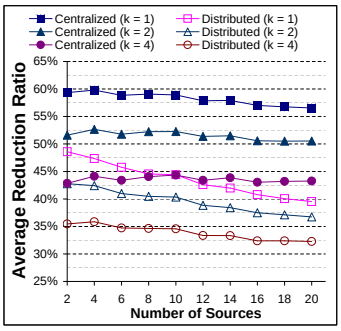


Fig. 11. Graph of the average reduction ratio of the centralized and distributed GG+INS+FO optimizations as a function of the number of sources, a data chunk of 75MB and different values of k.

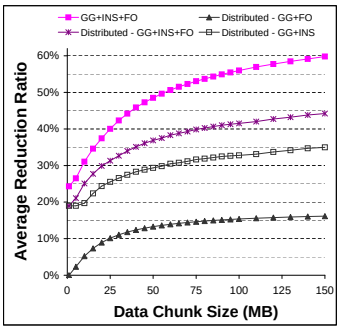


Fig. 12. Graph of the average reduction ratio of the centralized optimization (INS+FO) and three distributed optimizations (FO, INS, INS+FO) as a function of data chunk size for the greedy geographic tree GG.

further but available relay nodes whose help is only offered to cheaper but nearby links. This problem is exacerbated as the data chunk size increases. We varied values for B between 10% and 50% and for R between 1 and 3. For all combinations of B and R that we tested, we obtained similar results to those of Fig. 12. As in the centralized case, distributed INS is more effective than distributed FO. However, doing both distributed optimizations does result in roughly a 10% improvement compared to only doing the distributed INS optimization for most data chunk sizes.

Similar to the centralized implementation, we observe a slow reduction in the improvement ratio as the number of sources increases for $k = 2$ and 4 (Fig. 11). For cheaper mobility cost ($k = 1$), the difference in improvement ratios increases at a faster rate and reaches 9% as the number of sources increases from 2 to 20. This is because when mobility is cheaper, in an optimal setting, nodes can move over longer distances to help expensive links. However, as we mentioned earlier, in a distributed setting, mobile nodes are not aware of those distant expensive edges. Moreover, as the number of sources increases, the number of mobile nodes available to help decreases. Both factors combined make the distributed implementation slightly less effective for a high number of sources.

9 Conclusion

In this paper, we proposed a holistic approach to minimize the total energy consumed by both mobility of relays and wireless transmissions. Most previous work ignored the energy consumed by moving mobile relays. When we model both sources of energy consumption, the optimal position of a node that receives data from one or multiple neighbors and transmits it to a single parent is not the midpoint of its neighbors; instead, it converges to this position as the amount of data transmitted goes to infinity. Ideally, we start with the optimal initial routing tree in a static environment where no nodes can move. However, our approach can work with less optimal initial configurations including one generated using only local information such as greedy geographic routing. Our approach improves the initial configuration using two iterative schemes. The first inserts new nodes into the tree. The second computes the optimal positions of relay nodes in the tree given a fixed topology. This algorithm is appropriate for a variety of data-intensive wireless sensor networks. It allows some nodes to move while others do not because any local improvement for a given mobile relay is a global improvement. This allows us to potentially extend our approach to handle additional constraints on individual nodes such as low energy levels or mobility restrictions due to application requirements. Our approach can be implemented in a centralized or distributed fashion. Our simulations show it substantially reduces the energy consumption by up to 45%.

Acknowledgements

The authors thank Philip McKinley, Chiping Tang and Sandeep Kulkarni for their help.

References

[1] R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson, and D. Culler, "An analysis of a large scale habitat monitoring application," in *SenSys*, 2004.

[2] L. Luo, Q. Cao, C. Huang, T. F. Abdelzaher, J. A. Stankovic, and M. Ward, "Enviromic: Towards cooperative storage and retrieval in audio sensor networks," in *ICDCS*, 2007, p. 34.

[3] D. Ganesan, B. Greenstein, D. Perelyubskiy, D. Estrin, and J. Heidemann, "An evaluation of multi-resolution storage for sensor networks," in *SenSys*, 2003.

[4] S. R. Gandham, M. Dawande, R. Prakash, and S. Venkatesan, "Energy efficient schemes for wireless sensor networks with multiple mobile base stations," in *Globecom*, 2003.

[5] J. Luo and J.-P. Hubaux, "Joint mobility and routing for lifetime elongation in wireless sensor networks," in *INFOCOM*, 2005.

[6] Z. M. Wang, S. Basagni, E. Melachrinoudis, and C. Petrioli, "Exploiting sink mobility for maximizing sensor networks lifetime," in *HICSS*, 2005.

[7] A. Kansal, D. D. Jea, D. Estrin, and M. B. Srivastava, "Controllably mobile infrastructure for low energy embedded networks," *IEEE Transactions on Mobile Computing*, vol. 5, pp. 958–973, 2006.

[8] G. Xing, T. Wang, W. Jia, and M. Li, "Rendezvous design algorithms for wireless sensor networks with a mobile base station," in *MobiHoc*, 2008, pp. 231–240.
[9] D. Jea, A. A. Somasundara, and M. B. Srivastava, "Multiple controlled mobile elements (data mules) for data collection in sensor networks," in *DCOSS*, 2005.
[10] R. Shah, S. Roy, S. Jain, and W. Brunette, "Data mules: Modeling a three-tier architecture for sparse sensor networks," in *IEEE SNPA Workshop*, 2003.
[11] S. Jain, R. Shah, W. Brunette, G. Borriello, and S. Roy, "Exploiting mobility for energy efficient data collection in wireless sensor networks," *MONET*, vol. 11, pp. 327–339, 2006.
[12] W. Wang, V. Srinivasan, and K.-C. Chua, "Using mobile relays to prolong the lifetime of wireless sensor networks," in *MobiCom*, 2005.
[13] D. K. Goldenberg, J. Lin, and A. S. Morse, "Towards mobility as a network control primitive," in *MobiHoc*, 2004, pp. 163–174.
[14] A. A. Somasundara, A. Ramamoorthy, and M. B. Srivastava, "Mobile element scheduling with dynamic deadlines," *IEEE Transactions on Mobile Computing*, vol. 6, pp. 395–410, 2007.
[15] Y. Gu, D. Bozdag, and E. Ekici, "Mobile element based differentiated message delivery in wireless sensor networks," in *WoWMoM*, 2006.
[16] K. Dantu, M. Rahimi, H. Shah, S. Babel, A. Dhariwal, and G. S. Sukhatme, "Robomote: enabling mobility in sensor networks," in *IPSN*, 2005.
[17] http://www.k-team.com/robots/khepera/index.html.
[18] J.-H. Kim, D.-H. Kim, Y.-J. Kim, and K.-T. Seow, *Soccer Robotics*. Springer, 2004.
[19] G. Xing, T. Wang, Z. Xie, and W. Jia, "Rendezvous planning in wireless sensor networks with mobile elements," *IEEE Transactions on Mobile Computing*, vol. 7, pp. 1430–1443, 2008.
[20] ——, "Rendezvous planning in mobility-assisted wireless sensor networks," in *RTSS '07: Proceedings of the 28th IEEE International Real-Time Systems Symposium*, 2007, pp. 311–320.
[21] C.-C. Ooi and C. Schindelhauer, "Minimal energy path planning for wireless robots," in *ROBOCOMM*, 2007, p. 2.
[22] C. Tang and P. K. McKinley, "Energy optimization under informed mobility," *IEEE Trans. Parallel Distrib. Syst.*, vol. 17, pp. 947–962, 2006.
[23] E. D. Demaine, M. Hajiaghayi, H. Mahini, A. S. Sayedi-Roshkhar, S. Oveisgharan, and M. Zadimoghaddam, "Minimizing movement," in *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, ser. SODA '07, 2007, pp. 258–267.
[24] O. Tekdas, Y. Kumar, V. Isler, and R. Janardan, "Building a communication bridge with mobile hubs," in *Algorithmic Aspects of Wireless Sensor Networks*, S. Dolev, Ed. Springer-Verlag, 2009, pp. 179–190.
[25] Y. Mei, Y.-H. Lu, Y. Hu, and C. Lee, "Deployment of mobile robots with energy and timing constraints," *Robotics, IEEE Transactions on*, vol. 22, no. 3, pp. 507 – 522, june 2006.
[26] A. Sipahioglu, G. Kirlik, O. Parlaktuna, and A. Yazici, "Energy constrained multi-robot sensor-based coverage path planning using capacitated arc routing approach," *Robot. Auton. Syst.*, vol. 58, pp. 529–538, May 2010.
[27] M. Karpinski and A. Zelikovsky, "New approximation algorithms for the steiner tree problems," *J. Comb. Optim.*, vol. 1, no. 1, pp. 47–65, 1997.
[28] G. Robins and A. Zelikovsky, "Tighter bounds for graph steiner tree approximation," *SIAM J. Discrete Math.*, vol. 19, no. 1, pp. 122–134, 2005.
[29] S. Arora, "Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems," *J. ACM*, vol. 45, pp. 753–782, September 1998.
[30] K. Jain and V. V. Vazirani, "Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation," *J. ACM*, vol. 48, pp. 274–296, March 2001.
[31] M. Mahdian, Y. Ye, and J. Zhang, "Improved approximation algorithms for metric facility location problems," in *Proceedings of the 5th International Workshop on Approximation Algorithms for Combinatorial Optimization*, ser. APPROX '02, 2002, pp. 229–242.
[32] L. Wang and Y. Xiao, "A survey of energy-efficient scheduling mechanisms in sensor networks," *Mob. Netw. Appl.*, vol. 11, pp. 723–740, 2006.
[33] G. Wang, M. J. Irwin, P. Berman, H. Fu, and T. F. L. Porta, "Optimizing sensor movement planning for energy efficiency," in *ISLPED*, 2005, pp. 215–220.
[34] "Cc2420 datasheet," http://inst.eecs.berkeley.edu/cs150/Documents/CC2420.pdf.
[35] "Cc1000 single chip very low power rf transceiver." http://focus.ti.com/lit/ds/symlink/cc1000.pdf.
[36] M. Sha, G. Xing, G. Zhou, S. Liu, and X. Wang, "C-mac: Model-driven concurrent medium access control for wireless sensor networks," in *INFOCOM*, 2009.
[37] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *HICSS*, 2000.
[38] S. Ratnasamy, B. Karp, S. Shenker, D. Estrin, R. Govindan, L. Yin, and F. Yu, "Data-centric storage in sensornets with ght, a geographic hash table," *MONET*, vol. 8, pp. 427–442, 2003.
[39] C. E. Perkins and E. M. Royer, "Ad-hoc on-demand distance vector routing," in *WMCSA '99: Proceedings of the Second IEEE Workshop on Mobile Computer Systems and Applications*. Washington, DC, USA: IEEE Computer Society, 1999, p. 90.

Fatme El-Moukaddem received the BS and MS degrees in Computer Science from the American University of Beirut in 2000 and 2002 respectively. She is pursuing her PhD at Michigan State University. Her research interests include algorithms and wireless sensor networks.

Eric Torng received the Ph.D. degree in computer science from Stanford University, Stanford, CA, in 1994. He is currently an Associate Professor and Graduate Director with the Department of Computer Science and Engineering, Michigan State University, East Lansing. His research interests include algorithms, scheduling, and networking. Dr. Torng received a National Science Foundation CAREER Award in 1997.

Guoliang Xing received the B.S. degree in electrical engineering and the M.S. degree in computer science from Xi'an Jiao Tong University, China, in 1998 and 2001, respectively, and the M.S. and D.Sc. degrees in computer science and engineering from Washington University in St. Louis, in 2003 and 2006, respectively. He is an Assistant Professor in the Department of Computer Science and Engineering at Michigan State University. From 2006 to 2008, he was an Assistant Professor of Computer Science at City University of Hong Kong. He received the NSF CAREER Award in 2010. His research interests include wireless sensor networks, mobile systems, and cyber-physical systems. He received the Best Paper Award at the 18th IEEE International Conference on Network Protocols (ICNP), 2010.