MobiRoute: Routing towards a Mobile Sink for Improving Lifetime in Sensor Networks

Jun Luo Jacques Panchard Michał Piórkowski
Matthias Grossglauser Jean-Pierre Hubaux
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne (EPFL), CH-1015 Lausanne, Switzerland
Email: *firstname.lastname@epfl.ch*

Abstract

Improving network lifetime is a fundamental challenge of wireless sensor networks. One possible solution consists in making use of mobile sinks. Whereas theoretical analysis shows that this approach does indeed benefit network lifetime, practical routing protocols that support sink mobility are still missing. In this paper, in line with our previous efforts, we investigate the approach that makes use of a mobile sink for balancing the traffic load and in turn improving network lifetime. We engineer a routing protocol, MobiRoute, that effectively supports sink mobility. Through intensive simulations in TOSSIM with a mobile sink and an implementation of MobiRoute, we prove the feasibility of the mobile sink approach by demonstrating the improved network lifetime in several deployment scenarios.

I. Introduction

Many proposals on using mobile sinks to improve the lifetime of *wireless sensor network*s (WSNs) have appeared recently [1, 2, 3, 4, 5, 6, 7, 8, 9]. However, in the research community there is a doubt that moving sinks is practical (e.g., [10]). One of the major concerns behind this doubt is that mobility inevitably incurs additional overhead in data communication protocols and the overhead can potentially offset the benefit brought by mobility. In this paper, we intend to dismiss the doubt.

We focus on a scenario where all nodes are fixed and have limited energy reserves and where a mobile sink endowed with significantly more resources serves as the data collector. In this scenario, the sink mobility can increase network lifetime through two different methods, depending on the relationship between the sink moving *speed* and the tolerable *delay* of the data delivery.

In the *fast mobility* regime, the speed produces tolerable data delivery delay. The WSNs may then take advantage of *mobility capacity* [11]. This *mobile relay* approach [1, 2, 3] uses the mobile sink to transport data with its mechanical movements. It trades data delivery latency for the reduction of node energy consumption. We refer to [3] and [4] for simulations and field studies in this regime. In the *slow mobility* regime, the sink mobility takes a **discrete** form: the movement trace consists of several *anchor point*s between which the sink moves and at which it pauses. Consequently, the network cannot benefit from mobility capacity. However, it has recently been observed [5, 6, 7, 8, 9] that sink mobility can still improve network lifetime. The reason is that the typical many-to-one traffic pattern in WSNs imposes a heavy forwarding load on the nodes close to sinks. While no energy conserving protocol alleviates such a load, moving the sink (even very **infrequently**) can distribute over time the role of bottleneck nodes and thus even out the load. Unfortunately, theoretical analysis [5, 6, 7, 8, 9] may produce misleading results due to its simplified system model (an example is given in Section V: footnote 4); simulations involving a detailed protocol implementation are necessary to fully understand the benefit of using mobile sinks.

We argue that the slow mobility conditions exist in many realistic applications of WSNs. For example, suppose that a WSN is equipped with batteries that cannot be replaced, e.g., because the sensor nodes are

The work presented in this paper was supported (in part) by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322. (`http://www.terminodes.org`)

not accessible, or because changing batteries would be hazardous or costly. This may be the case for sensors in smart buildings, where batteries might be designed to last for decades, and for environmental or military sensing under hostile or dangerous conditions (e.g., avalanche monitoring). In this case, it may be desirable and comparatively simple to move a sink infrequently (e.g., once a day or a week) by a human or by a robot. For example, in the avalanche monitoring scenario, a sink may be deployed at the periphery[1] of the monitored area and then moved once in a while by a helicopter. In the building scenario, the sink can be "virtually" moved: computers in different offices serve as the sink in shifts. In the military surveillance scenario, moving the sink may require some effort, but it can be acceptable if done infrequently.

Numerous routing protocols have been proposed in the last decade to support data communications in either mobile ad hoc networks (MANETs) or WSNs. On one hand, the protocols for MANETs (e.g., [12, 13]) are definitely an overkill for supporting mobile sinks because the basic assumption in MANETs is that every node moves in an unpredictable way. On the other hand, the protocols for WSNs usually take static sinks for granted (e.g., [14, 15], although some exceptions exist [16, 17, 18, 3]). It is true that a routing protocol that supports mobile sinks to collect data in WSNs, compared with existing protocols for static WSNs, will have higher protocol complexity and overhead. However, the following favorable features of the sink mobility (in the low mobility regime) and of the existing routing protocols help to limit the side effects:

- The mobility is controllable and thus predictable,
- The pause time of a sink along its moving trace is much longer than the actual moving time,
- Existing routing protocols usually possess some proactive features to cope with the dynamics in link quality; this can be exploited to support sink mobility.

In the spirit of [8] (where we theoretically prove the superiority of mobile sinks over static ones), we further investigate in this paper the performance, with respect to both lifetime and reliability (measured by packet delivery ratio), of WSNs with a mobile sink. We consider a scenario where nodes periodically transfer data through multi-hop routes towards the sink, while the sink intermittently changes its position according to certain predefined traces. We propose a routing protocol, MobiRoute, dedicated to support sink mobility. It takes into account the favorable features we mentioned above and thus only marginally increases the protocol complexity and overhead. We use TOSSIM [19] as the simulator. Our simulation results demonstrate the efficiency of MobiRoute, in terms of both an improved network lifetime and an undegraded reliability, in several deployment scenarios. Our contribution with respect to [5, 6, 7, 8, 9] are:

- Our investigation is based on a practical routing protocol. Consequently, we take into account realistic conditions such as control overhead with a routing protocol and collision/overhearing [16, 20] at the MAC layer.
- We also look at the reliability issue incurred by sink mobility, which is not considered by the numerical simulations of the previous work.

The rest of this paper is organized as follows: Section II surveys related work. Section III clarifies our metrics and methodologies. Section IV presents our MobiRoute protocol. Section V describes the algorithm that controls the sink mobility in an adaptive way. Simulation results are reported in Section VI. Finally, Section VII concludes the paper.

II. Related Work

In this section, we briefly survey the existing routing protocols that are designed to support sink mobility. Whereas the previous proposals [5, 6, 7, 8, 9] serve as the theoretical basis of this paper, we will not discuss them further because this paper focuses on the practicality of sink mobility instead of theoretical analysis. We refer to Section I for the principle of these proposals.

There are a few proposals for data collection with mobile sinks [16, 17, 18, 3]. While most of them [16, 17, 18] consider sink mobility as an inherent behavior of WSNs and try to **cope with** it, only [3] share the same opinion of **exploiting** controllable sink mobility as ours.

[1] As we have shown in [8], the optimum trace (in terms of network lifetime) for a mobile sink is the network periphery.

The *two-tier data dissemination* (TTDD) approach [16] is actually based on the existing idea of virtual backbone. The backbone (or grid in TTDD's terminology) is formed proactively upon detection of a stimulus. The mobile sinks send queries to the nearest grid points with a flooding. Queries are routed along the grid and data trace the reverse path back to the sinks. As a consequence, the control overhead introduced by sink mobility is limited to the grid cell where a sink is located. Aiming at further limiting the widespread diffusion of control messages introduced by sink mobility, the *scalable energy-efficient asynchronous dissemination* (SEAD) protocol [17] assigns particular nodes as the *access node*s and relies on such fixed anchors to limit the control traffic. Mobile sinks only need to select one of their neighbors as an access node and maintain the link with it. SEAD constructs an energy-efficient dissemination tree from a source to difference access nodes. Data are routed along the tree and then unicast from the access nodes to their sinks. To handle mobility, sink may handoff from one access node to the other if the tradeoff between the energy consumed to build another tree and the data delivery latency exceeds a given threshold. Both TTDD and SEAD heavily rely on the assumption of location-aware sensor nodes, which are not required in our case. Moreover, the data generation model is also different from ours. Under our data generation model (where every node produces data with the same rate), the virtual backbone used in TTDD to carry most traffic would become fixed and the delayed handoff in SEAD could lead to suboptimal routing trees for a substantial amount of time; they both offset the load balancing effect resulting from sink mobility.

The *hybrid learning-enforced time domain routing* (HLETDR) proposed in [18] comes a bit closer to our situation. They assume a somewhat fixed sink trace and the same data generation model as ours. However, instead of requiring the sink to inform other nodes about its location changes, HLETDR lets sensor nodes to figure out the sink trace through a learning-based approach. This learning process is done through positive and negative reinforcements according to the probability density function indicating how far the sink is from the nodes close to the sink trace. Tour period of the sink is divided into m domains. When a node has data to be forwarded, the probability of selecting next hop is determined according to its possibility being on a shortest path to the sink in that time domain. HLETDR has a high complexity of the routing table, i.e., $O(m)$, when a fine time granularity is required. In addition, the relatively slow learning procedure limits the adaptability of the sink mobility (which is very crucial for load balancing in our case).

The routing protocol described in [3] is based on *directed diffusion* [14], a routing protocol dedicated to *data-centric*[2] communications. [3] extends directed diffusion by adding mainly three components: 1) a pre-move phase for the nodes to learn the sink trace, 2) an acknowledgement-retransmit scheme to handle packets loss during handoff, and 3) a pre-fetch mechanism to improve the data delivery. The data generation model assumed in [3] is quite different from ours: nodes send data only when the data are queried whereas nodes proactively push data to the sink in our model. In addition, the sink moves continuously in [3] because their protocols work in the fast mobility regime (see Section I for details). Therefore, the control objective is the moving speed for [3], whereas we consider the adjustment of pause times. In this paper, we take an approach similar to that of [3] in that, instead of developing a routing protocol from scratch, we extend an existing routing protocol with the ability of handling sink mobility.

III. Problem, Metrics and Methodology

We define network lifetime as the time period for the first node to run out of its energy reserve [21]. When evaluating this quantity, we convert the problem of maximizing network lifetime to a min-max problem in terms of the **radio** energy consumption of individual nodes. Another performance index we want to evaluate is the packet delivery ratio (or *reliability*). In fact, a possible side-effect brought by sink mobility could be an increase in packet loss due to occasional topology changes; the lifetime elongation resulting from sink mobility is justifiable only if the increase in packet loss is tolerable.

We assume that nodes generate data and send them to the sink with the same rate. In our approach, the mobility pattern of a sink takes a *discrete* form [6]: the moving trace consists of several *anchor point*s

[2]Data generated by sensor nodes are named by attribute-value pairs. A node requests data by sending interests for named data rather than for named nodes.

between which the sink moves and at which the sink pauses. We require each *epoch* (the time during which the sink pauses) to be much longer than the moving time, such that the routing overhead introduced by sink mobility becomes negligible due to its amortization across a long epoch. Imposing these anchor points simplifies the design of the mobile sink[3] and limits the extra overhead introduced to the routing protocol (see Section IV for details). In addition, a continuous movement is not necessary, as a granularity of (sink) displacement smaller than the magnitude of the effective radio range may not lead to any topological change (whereas topological changes are what we expect from the sink mobility). In order to better adapt to the topology and dynamics of a given network, we also intend to control the sink mobility *on-line* (based on the *off-line* optimization described in [8]).

Our experiment methodology involves simulations with TOSSIM [19]. The main benefit of using the TOSSIM simulator is that the protocol used for simulations can be directly adopted by real sensor nodes. We simulate a set of networks with nodes on 4×4, 5×5, and 7×7 point lattices; these scenarios represent outdoor WSNs in general. We also simulate a network that we intend to deploy as an in-building testbed.

IV. MobiRoute: Routing Towards a Mobile Sink

According to the definition of discrete mobility pattern described in Section III, the sink changes its location from time to time. A routing protocol that transfers data towards such a sink should perform the following operations that are not needed for traditional WSNs:

1) Notify a node when its link with the sink gets broken due to mobility.
2) Inform the whole network of the topological changes incurred by mobility.
3) Minimize the packet loss during the sink moving period.

Operation 1 seems to be encompassed by 2, but the level of urgency is different. Packets forwarded by a last-hop node will get lost if the node does not detect the link breakage, while a remote node can still send its data to the sink successfully without knowing the topological changes. However, the routing optimality is compromised without operation 2. It is not possible to avoid packet loss, because a realistic failure detector (which usually relies on a timer) always has some delay. Therefore, the goal of operation 3 is to minimize rather than eliminate packet loss. Possible scenarios related to these operations are illustrated in Fig. 1.

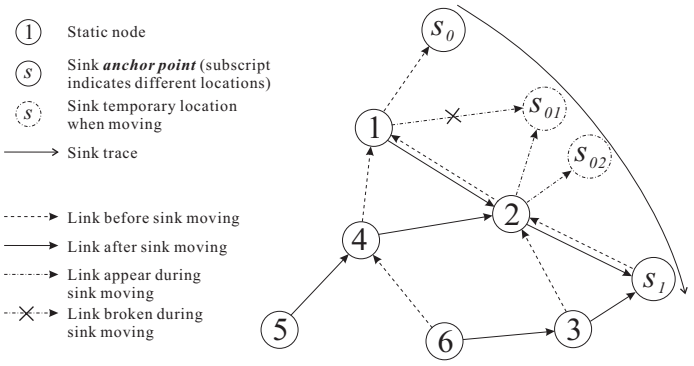


Fig. 1. This example illustrates possible scenarios where additional operations are necessary. Assuming the sink, after its (long) pause at s_0, moves to s_1, (1) the link breakage happening when the sink reaches intermediate location s_{01} (where it loses connectivity with node 1) should be notified to node 1, otherwise the node will have to drop packets sent from other nodes, (2) nodes 3, 4, and 6 should be informed about the topological changes at a proper time, otherwise, for example, 6 might take the following sub-optimal routing path: $6 \rightarrow 4 \rightarrow 1 \rightarrow 2 \rightarrow s$.

Our routing protocol, MobiRoute, is a superset of Berkeley MintRoute [15]. MobiRoute extends MintRoute by adding functions that perform the aforementioned operations. We first introduce MintRoute briefly in Section IV-A, then we describe the extended functions of MobiRoute separately in Sections IV-B to IV-D. The state diagram shown in Fig. 2 is used when we present MobiRoute.

[3]The mobile sink can simply be a laptop (moved occasionally by a human), rather than a sophisticate robot as used in [3].

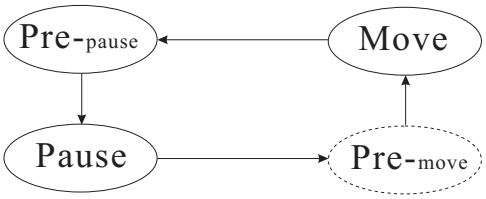


Fig. 2. States and transitions involved in MobiRoute. Note that only the protocol running at the sink side has the *pre-move* state.

A. MintRoute

Berkeley MintRoute [15] is a routing protocol designed specifically for the all-to-one data transmission style of WSNs. It takes a distributed distance-vector based approach: route messages (i.e., control packets) are exchanged periodically among neighbor nodes, and the next hop nodes (or *parent*s in MintRoute nomenclature) are chosen by evaluating the costs of routing data through different neighbors. The exchanged route messages not only help to measure the distance (in terms of the number of possible transmissions) from the sink but also provide a way to evaluate the link qualities (from both directions) between nodes. As a result, MintRoute applies a **M**inimum **T**ransmission (MT) metric, where the goal is to minimize the total number of transmissions (including retransmissions). Since the data rate in WSNs is low, route messages do not need to be exchanged frequently (the rate is actually a multiple of the data rate in MintRoute). This helps MintRoute to reduce its energy consumption. Although MintRoute does not explicitly apply a metric that considers load balancing, the protocol, according to our experience, balances the traffic load with occasional switches of nodes' parents (which is a direct consequence of the MT metric). This feature makes MintRoute a leading candidate for supporting sink mobility. Finally, MintRoute applies a sequence number for each packet to detect packet loss and thus evaluate link quality; this sequence is shared by both control and data packets.

B. Detecting Link Breakage

In order to inform the nodes located close to the sink trace about the state of their links with the sink, MobiRoute applies a beacon mechanism. The sink, during the whole moving period, periodically broadcasts a beacon message (*s-beacon* hereafter). A node, upon receiving a s-beacon, sets (or resets) its **detecting** timer. If the timer times out before receiving the next s-beacon, the failure detector at this node indicates a link breakage and a new parent is chosen. We now discuss several crucial points of this seemingly simple mechanism.

First, we require the sink to transit from the **pause** state to the **pre-move** state before physically beginning to move. The sink begins to broadcast s-beacons under the pre-move state and evolves to the **move** state after a while. The sink moves while broadcasting s-beacons under the move state. A node, after receiving the first s-beacon under its current **pause** state, transits to the **move** state directly. Nevertheless, the pre-move state (of the sink) is necessary: it guarantees the reception of s-beacons at the nodes' side before the link quality changes due to the sink mobility.

Secondly, although only the sink (whose energy reserve is abundant) spends energy to send s-beacons, nodes also spend energy to receive these beacons. Therefore, the frequency of s-beacons should not be too high. On the other hand, low frequency sending retards failure detection, which in turn increases packet loss. We apply a simple heuristic: the frequency is set in the same order as the accumulative packet sending rate. For example, if the sending rate of each node is 1 pkt/min in a 60-node network, the accumulative rate at a last-hop node is at most 1 packet/second, and the beacon frequency is set to 1Hz. A related parameter is the timeout value for the detecting timer. Fortunately, the value can be relatively small, because a node will detect a false-positive when receiving another s-beacon.

Finally, the beacon mechanism is a costly procedure, regardless of which beacon frequency is chosen. Fortunately, since the moving period accounts only for a small fraction of the network lifetime, its costs

will be amortized across the lifetime. A continuous sink movement, on the contrary, would incur such costs permanently.

C. Conveying Topological Changes

MobiRoute could have relied on MintRoute to propagate the topological changes resulting from sink mobility. However, the rate of route message exchanges in MintRoute is very low. Therefore, it takes a long time to convey the topological changes to the whole network; during this period, many packets are routed through sub-optimal paths, which consumes additional energy and thus offsets the benefit of sink mobility. As a result, MobiRoute needs a speed-up (route message exchange) rate for propagating the topological changes.

Propagating information throughout a network is a costly procedure (message complexity $O(n)$); it cannot be performed frequently. So MobiRoute only performs a propagation upon the sink reaching an anchor, and it tolerates a limited number of sub-optimal routing during the moving period. The sink enters the **pre-pause** state (see Fig. 2) when it stops moving; it then sends route messages with a speed-up rate, which causes their receivers to enter the same state. Nodes that receive messages **directly** from the sink also send speed-up route messages; they re-evaluate the quality of their links with the sink using these exchanges. A node receiving speed-up messages **indirectly** also enters the pre-pause state; it forwards the message only if its distance towards the sink changes significantly (e.g., node 6 in Fig. 1 might not forward messages received from node 3). The energy consumption of the propagation procedure is effectively reduced, because there are nodes that are not affected for a given move of the sink. Every node (including the sink) in the pre-pause state transits to the **pause** state after a short time span controlled by a timer.

D. Minimizing Packet Losses

Although packet loss cannot be avoided during the sink moving period due to the lag of the failure detector, there are ways to mitigate the losses. Taking advantage of having a very short moving period (which we would not have if the sink moved continuously), the protocol tries to reduce the sending rate of the last-hop nodes, by asking them to buffer data packets using the interface queue (`QueuedSend` module) in MintRoute. We also add the following command to `QueueControl` interface, such that the routing module

```
command void QueueControl.setAddrInQueue(uint16_t parent)
{
    uint16_t i;
    if (!fQueueIdle)
        for (i = dequeue_next; i != enqueue_next;
             i = (i + 1) % MESSAGE_QUEUE_SIZE)
            msgqueue[i].address = parent;
}
```

can access the interface queue to change the next-hop address of the buffered packets upon detecting a link failure.

Nodes can only buffer data packets; control packets should still be sent. However, if we simply picked up control packets from the interface queue and sent them, there would be gaps among the sequence numbers (remember that MintRoute applies the same sequence for both control and data packets). These gaps would mislead a neighbor node about a degradation in the link quality. Two solutions can be applied: 1) using separate sequences and queues for data and control packets or 2) rearranging the sequence number within the queue, such that packets sent have consecutive sequence numbers. In the short term, we adopt the second solution because it is easy to implement, but the first could be desirable in a long-run perspective.

V. Adaptively Controlled Mobility

According to our simulation results in Section VI, a mobility strategy that adapts to the network topology (for which no a priori knowledge exists) performs better than a static schedule. In this section, we describe the adaptive algorithm to control sink mobility. Our algorithm adaptively changes the epoch of the sink at each anchor point, according to the power consumption profile of the network. We derive the algorithm from the following linear program:

$$\text{Maximize} \quad \text{lifetime } T = \sum_k t_k \tag{1}$$

$$\text{Constraints}: \quad \sum_k t_k \mathbb{P}_k \leq \mathbb{E} \tag{2}$$

where $\mathbb{P}_k$ and $\mathbb{E}$ are vectors that represent the power consumptions of each node (referred to as *P-profile* hereafter) when the sink pauses at a certain anchor point k and the initial energy reserves of all nodes, respectively. This formulation basically means that we weigh, through the epoch t_k, the anchor points based on the corresponding P-profile $\mathbb{P}_k$s, in such a way that the $\mathbb{P}_k$s that complement each other are favored. Although this LP formulation is similar to what is described in [6, 7], our contribution lies in the fact that we define the $\mathbb{P}_k$s by instrumenting the prototype of a routing protocol, while [6, 7] only manipulate flows on a graph[4].

In practice, we propose the following 2-phase algorithm to approximate the above programming problem:

- **Phase I–Initialization**: The mobile sink visits the anchor points one by one and pauses at each point for a short *sampling period*. During each sampling period, the sink collects the power consumption records from all nodes and builds a P-profile for that anchor point. At the end of this phase, the sink performs the programming (1) and drops an anchor point if its weight T_i is extremely low. It is not worth keeping such a point because its corresponding epoch is not long enough to amortize the routing overhead introduced by the sink mobility.
- **Phase II–Operation**: The mobile sink goes through the trajectory repetitively but only pauses at those chosen anchor points. At a given point k, the sink again collects power consumption information and builds a profile $\mathbb{P}_k$. Based on the new profile and previous profiles for other chosen points, the programming (1) is re-solved to deduce t_k. The actual epoch is computed as $\hat{t}_k = t_k/\delta$, where $\delta > 1$ is an integer. Applying the δ makes it possible for the sink to repeat the movement pattern several turns, which allows the sink to be more adaptive to the network dynamics.

We have the following remarks on the algorithm:

- If we make a discrete search over the whole surface covered by the network to obtain the anchor points, the time to finish Phase I could become comparable to the network lifetime; the algorithm would thus lose its adaptability (e.g., the sink might not even get a chance to enter Phase II). Alternatively, we can search over a "good" trace. A candidate of such a trace could be the periphery of the network [8].
- The sink could have directly applied the results (i.e., t_ks) of the first phase to the second phase if the routing topology were fixed. However, according to our experiences with real WSNs, the routing topology keeps evolving even with static nodes. As a result, the P-profiles obtained from the first phase can only be considered as estimations and should be updated if new profiles are available.

VI. Simulations

We report two sets of simulations with TOSSIM in this section. In one set, network nodes are located on point lattices; simulation results of this set represent outdoor WSNs in general. In another set, nodes form a ring; the simulations emulate our future field tests with an in-building WSN.

[4]In fact, manipulating flow without bearing in mind the behaviors of a realistic routing protocol may leads to a misleading conclusion. For example, the formulation in [6] tries to maximize the lifetime by considering only a subset of flow (which is chosen to simplify the problem). However, this biased choice produces a significant deviation (in terms of the anchor points) from the theoretical optimum [22], on which our empirical settings (see Section VI) are based.

A. Grid Networks

We arrange nodes on a point lattice of size 4×4, 5×5, and 7×7. For each network, we either 1) put the sink (node 0) at the network border (the midpoint of one side), or 2) at the center, or 3) let the sink move around the network periphery. There is a constant distance between any two consecutive anchor points; the sink pauses on an anchor point and moves in between two anchors according to the instruction from a Tython [23] code. The connectivity[5] of a node with other nodes is shown in Fig. 3. The transmission range is set to

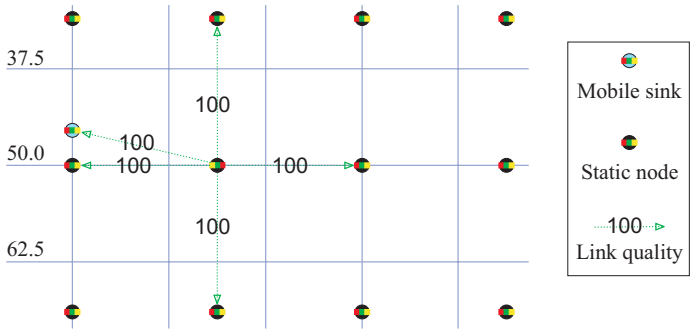


Fig. 3. Neighborhood graph in TinyViz [19]. The number beside a link states the link quality: 100 stands for a perfect link. The numbers on grid lines represent coordinates.

1.2 times longer than the distance between two neighbor nodes. Each node generates a data packet every 60 seconds. A control packet (route message) is sent every 120 seconds in the pause state and every 2 seconds (speed-up rate) in the pre-pause state. The s-beacon rate is one per second. The retransmission is disabled for all nodes if not stated otherwise. The epoch of non-adaptive mobility allows each node to send 10 data packets (i.e., 600 seconds)[6], and the moving time is 10 seconds for the 49 nodes network and 20 seconds for the other two. The sink moves at a speed of 1 ft/s in the move state. The full simulation time is just long enough to let the sink go through one round of its trip; the simulation for a given network is repeated 10 times. For the measurement of energy consumptions[7], we use the number of (both control and data) packets that a node is involved to characterize the energy consumption. By doing this, we implicitly assume that 1) radio communication is the dominating energy consumer, 2) sending and receiving a packet consumes the same amount of energy, and 3) control and data packets are of the same size.

1) Non-adaptive Mobility: The spatial distributions of energy consumptions for networks with 25 and 49 nodes are shown in Fig. 4. According to the lifetime definition in Section III, the smaller the maximum energy consumption in a network, the longer the network lifetime will be. Comparing the two cases with a static sink and the case with a mobile sink, we make the following observations:

- The load-balancing effect of using a mobile sink is evident. The network with a mobile sink always lives longer than the network with a static sink at its border and no shorter than the network with a static sink at the center.
- In the network of 49 nodes, using a mobile sink is the best choice, irrespective of whether the overhearing at the MAC layer exists or not. However, overhearing does offset the benefits of using a mobile sink: the 100% improvement on the lifetime (comparing the network having a mobile sink with the one having a centered static center) is reduced to 50% if overhearing exists.

[5]We take the *fixed radius* model, although it is less realistic than the *empirical* one (we refer to [19] for the definitions of these models). The reason is that, given a set of geo-distributed nodes, applying the empirical model usually leads to small network diameter due to the occasional existence of *shortcuts*. A relatively large network diameter (up to 10 hops) is essential to fully exhibit the benefit of using a mobile sink, but increasing the network size to achieve larger diameter results in a simulation time of unreasonable duration (e.g., 100 hours). By using the fixed radius model, we simply assume that, for a certain node, only nodes within its *effective region* [15] are considered as its neighbors.

[6]This duration is way shorter than what could be in a real deployment, where it might last for days or even weeks. Therefore, the performance of MobiRoute is expected to be better in practice, thanks to a longer amortization period.

[7]Since TOSSIM uses a MAC that never switches off its radio, tools such as Power-TOSSIM [24] always report a flat energy consumption pattern of a network no matter where the sink is located. In reality, motes equipped with B-MAC [20] do switch off their radio when there is no transmission going on.

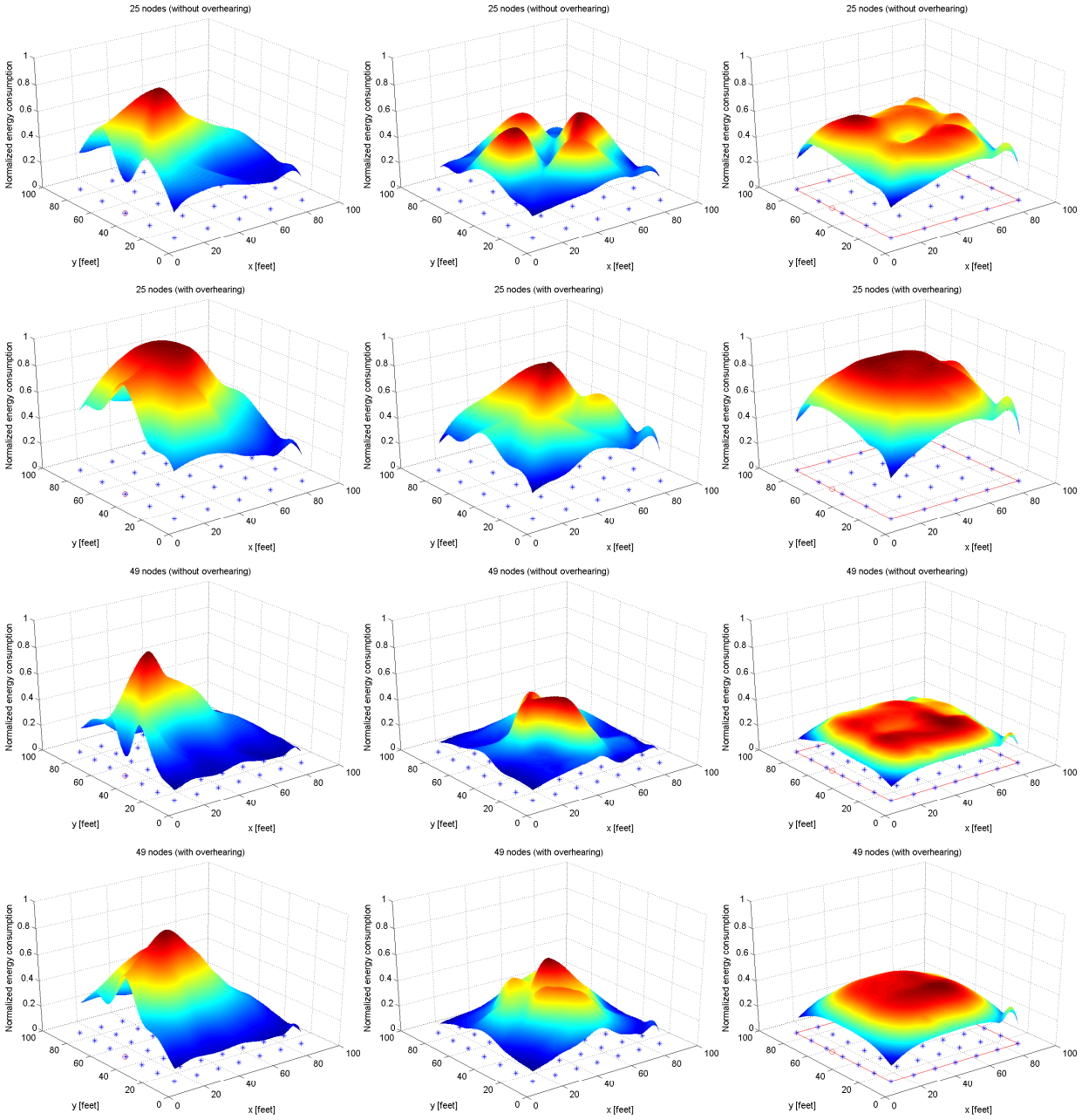


Fig. 4. Energy consumption of WSNs. Two networks with 25 and 49 nodes, respectively, are simulated. For each network, we either put the sink at the network border (the midpoint of one side), or at the center, or let the sink move around the network periphery. For each comparative case (i.e., one row in the figure), the energy consumptions are normalized to a common scale factor.

- In smaller networks of 16 and 25 nodes (only the latter case is shown in Fig. 4 due to their similarity), using a mobile sink is not necessarily helpful, because it does not improve the lifetime compared with using a centered static sink while increasing the accumulative energy consumption of the network.

A straightforward conclusion is that using a mobile sink is more beneficial in large networks. Since the function of the mobile sink is to disperse the traffic flows, the network should be large enough to provide nodes with a sufficient number of alternative routing paths. However, since locating a sink at the network center is not always practical[8], using a mobile sink does help to improve the lifetime in most networks.

[8]For habitat and environment monitoring, unobtrusive observation is key for studying natural phenomena [25]. Although nodes are small enough for this purpose, a sink (especially when it has to transmit the collected data out of the network area) can hardly makes itself invisible in the environment.

Another implication of our observations is that a MAC protocol free of overhearing is very important to improve the effectiveness of using a mobile sink. Unfortunately, the current MAC of motes (i.e., B-MAC [20]) suffers much from overhearing [25], and protocols with the potential to avoid overhearing (e.g., S-MAC [16]) do not necessarily have an overall performance better than B-MAC due to their burdensome synchronization schemes. So, we expect future technology to provide sensor nodes with overhearing-free MACs.

We plot the cumulative distribution functions of the packet delivery ratio in these two networks in Fig. 5. The comparisons are only made between a centered static sink and a mobile sink, because the ratios are quite similar for both networks with a static sink. The figures show that, without retransmission, the packet

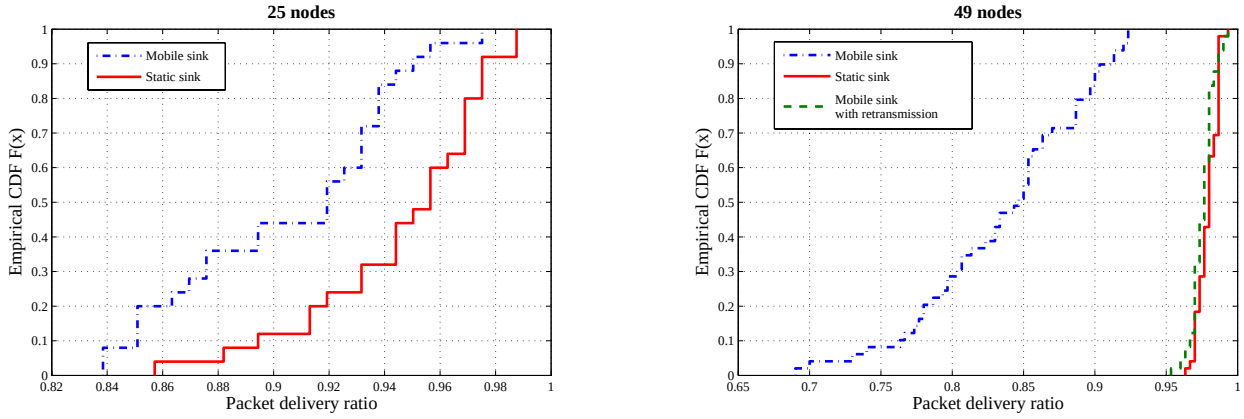


Fig. 5. Comparisons of packet delivery ratio

delivery ratio is always lower in the case of a mobile sink, which is intuitive (see the reasons that we described in Section IV). Also, the difference between the two ratios increases with the network size. The reason is that using a mobile sink increases the worst-case routing path length (actually, a static sink located at one vertex of the network periphery achieves the same ratio). This is not a major problem, because we would

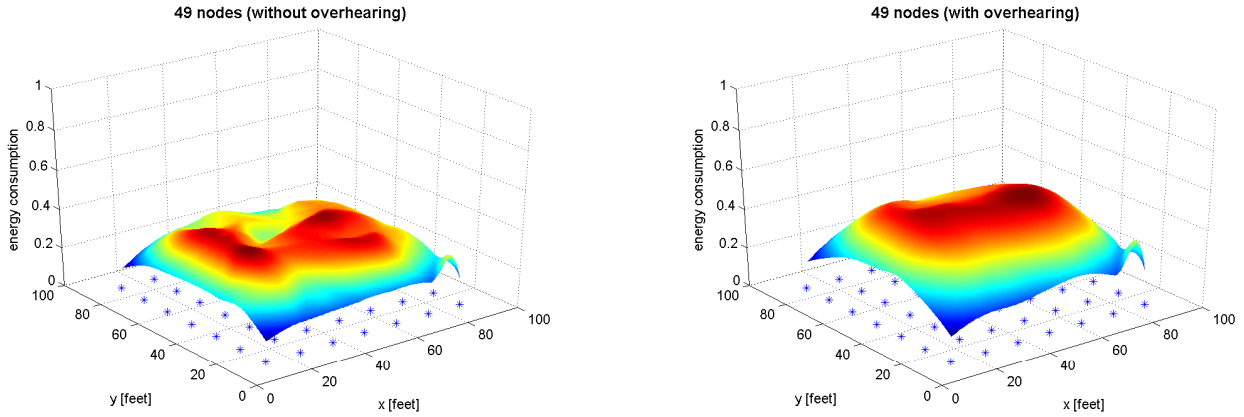
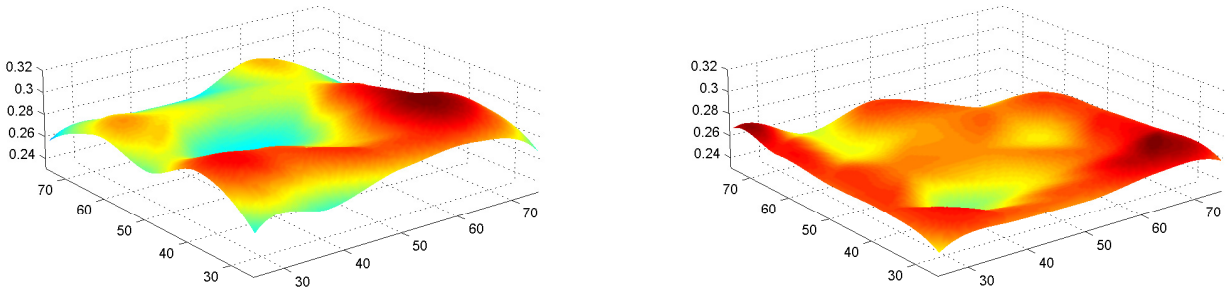


Fig. 6. Energy consumption of a WSN with a mobile sink and retransmission enabled. The scale factors take the same value as used for Fig. 4.

expect a much higher reliability in reality, where a node typically sends data only every tens of minutes [25]. Actually, if we enable the retransmission, the packet delivery ratio in the case of a mobile sink can be as high as that in the case of a static sink, but at the cost of increased energy consumption (Fig. 6), whose maximum value is still low enough to justify the benefit of using a mobile sink.

2) Adaptive Mobility: Zooming into the spatial distribution of energy consumption in the network with a mobile sink (as shown in Fig. 7 (a)), we observe that the load taken by nodes near the corner is heavier than that of other nodes. Applying the algorithm described in Section V, we actually find that the sink should



(a) Non-adaptive mobility

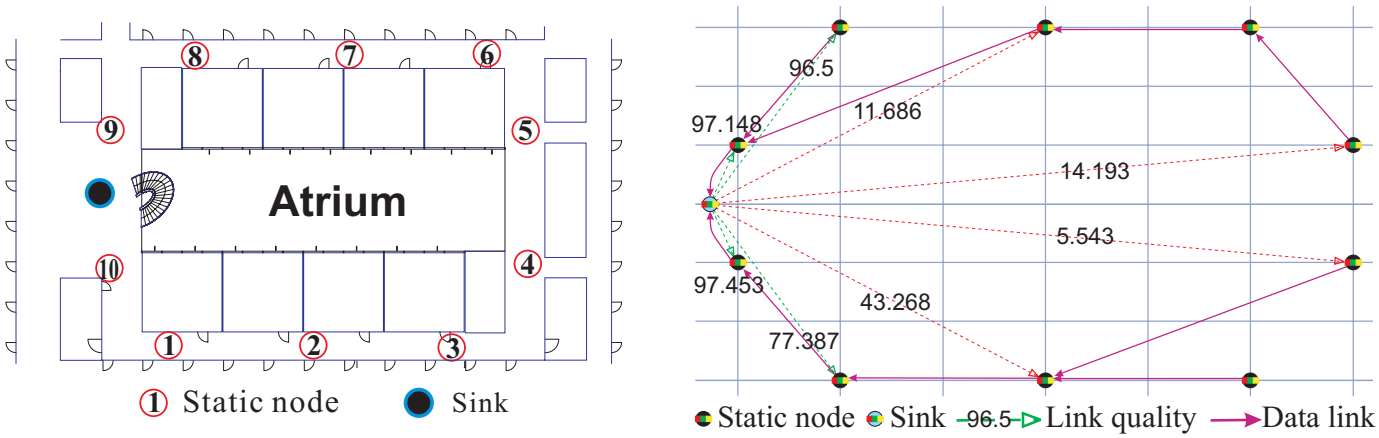
(b) Adaptive mobility

Fig. 7. Zooming in the distribution of energy consumption with a mobile sink.

pause less time at those anchors near the corner. The resulting load, shown in Fig. 7 (b), is further balanced; which improves the network lifetime by about 10%. Note that the sink, in our simulations, only circles around the network twice: one in phase I and another in phase II (see Section V); the network lifetime can be further improved with more rounds in phase II.

B. Ring Network

This section presents the simulation with a ring network. We use this simulation scenario to emulate a network deployed in our building, as shown in Fig. 8 (a). While a static sink[9] is located in-between nodes



(a) The experimental network

(b) Simulation scenario

Fig. 8. The plan of our network deployment (a) and the simulation scenario (b). Nodes are numbered the same way in (b) as in (a).

9 and 10, a mobile sink moves around the circle and pauses in between two consecutive nodes. We use the *empirical* model [19] to characterize the connectivity in this set of simulations. As an example, the connectivity graph for the sink (node 0) is shown in Fig. 3 (b). Each node generates a data packet every 30 seconds. A control packet (route message) is sent every 60 seconds in the pause state and every 2 seconds (speed-up rate) in the pre-pause state. The s-beacon rate is one per second. The retransmission is disabled for all nodes. Each of the 10 simulations lasts for 17600 seconds and the epoch of non-adaptive mobility is 1760 seconds. The sink moves at a speed of 1 ft/s in the move state, and the moving time is 25 seconds. The measurement of energy consumptions is the same as for Section VI-A, and the overhearing is not taken into account.

[9]The atrium inside of our building prevents us from locating the sink at its optimum position (i.e. the center of the network). This indeed corroborates our claim in Section VI-A that locating a sink at the network center is not always practical.

We illustrate the simulation results with bar graphs in Fig. 9. As shown in Fig. 9 (a), the load balancing effect is already very evident by simply moving an uncontrolled sink, which improves the lifetime by 20%. Further improvement is achieved (an additional 15% of improvement on lifetime compared to the non-adaptive mobility) by controlling the mobile sink adaptively. The behavior in packet delivery, plotted in

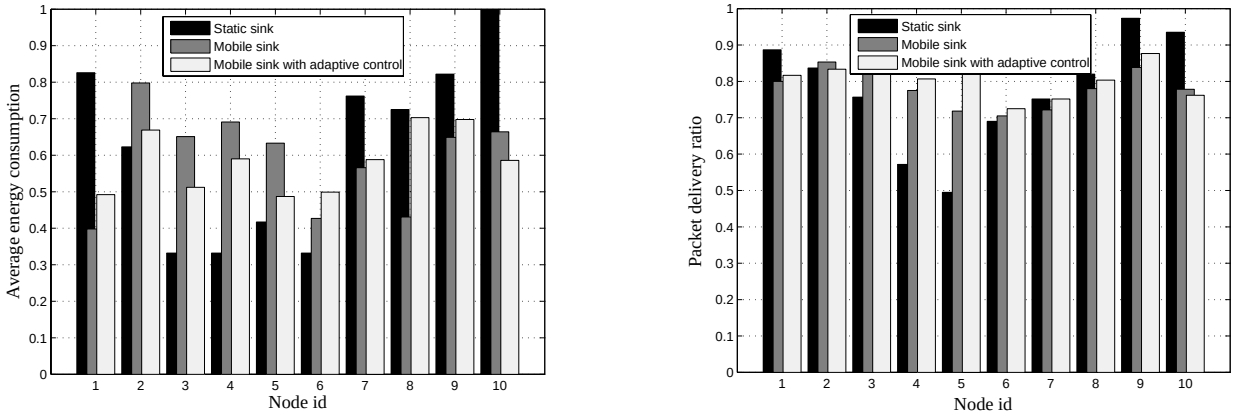


Fig. 9. Simulation results. (a) The energy consumptions are normalized by the largest energy consumption observed (i.e., node 10 in the case of a static sink). (b) The averaging effect arises also for the packet deliver ratio.

Fig. 9 (b), differs from that shown in Fig. 5; the averaging effect also arises due to the special network topology. In this specific scenario, the averaging effect makes a mobile sink beneficial not only to the network lifetime but also to the reliability, because nodes that are far away from the static sink perform poorly in terms of the reliability of packet delivery.

VII. Conclusion

In this paper, we have presented a routing protocol, MobiRoute, to support wireless sensor networks (WSNs) with a mobile sink. This is a follow-up of our previous work [8] where we theoretically proved that moving the sink can improve network lifetime without sacrificing data delivery latency. By intensively simulating MobiRoute with TOSSIM (in which real implementation codes are running), we have demonstrated the benefit of using a mobile sink rather than a static one. We have simulated both general networks with nodes located in point lattices and a special in-building network with nodes forming a ring. The results are very promising: a mobile sink, in most cases, improves the network lifetime with only a modestly degraded reliability in packet delivery.

We are in the process of performing full-scale field tests with the in-building network. We will also improve MobiRoute based on the experience obtained from our field tests.

References

[1] R.C. Shah, S. Roy, S. Jain, and W. Brunette. Data MULEs: Mobeling a Three-tier Architecutre for Sparse Sensor Networks. In *Proc. of the 1st IEEE SNPA*, 2003.

[2] A. Chakrabarti, A. Sabharwal, and B. Aazhang. Using Predictable Observer Mobility for Power Efficient Design of Sensor Networks. In *Proc. of the 2nd IEEE IPSN*, 2003.

[3] A. Kansal, A. Somasundara, D.D. Jea, M.B. Srivastava, and D. Estrin. Intelligent Fluid Infrastructure for Embedded Networks. In *Proc. of the 2nd ACM/USENIX MobiSys*, 2004.

[4] D. Jea, A. Somasundara, and M.B. Srivastava. Multiple Controlled Mobile Elements (Data Mules) for Data Collection in Sensor Networks. In *Proc. of the 1st IEEE/ACM DCOSS*, 2005.

[5] S.R. Gandham, M. Dawande, R. Prakash, and S. Venkatesan. Energy Efficient Schemes for Wireless Sensor Networks with Multiple Mobile Base Stations. In *Proc. of IEEE Globecom*, 2003.

[6] Z.M. Wang, S. Basagni, E. Melachrinoudis, and C. Petrioli. Exploiting Sink Mobility for Maximizing Sensor Networks Lifetime. In *Proc. of the 38th HICSS*, 2005.

[7] Z.M. Wang, E. Melachrinoudis, and S. Basagni. Voronoi Diagram-Based Linear Programming Modeling of Wireless Sensor Networks with a Mobile Sink. In *Proc. of the IIE Annual Conference and Exposition*, 2005.

[8] J. Luo and J.-P. Hubaux. Joint Mobility and Routing for Lifetime Elongation in Wireless Sensor Networks. In *Proc. of the 24th IEEE INFOCOM*, 2005.

[9] I. Papadimitriou and L. Georgiadis. Maximum Lifetime Routing to Mobile Sink in Wireless Sensor Networks. In *Proc. of the 13th IEEE SoftCom*, 2005.

[10] W. Wang, V. Srinivasan, and K.-C. Chua. Using Mobile Relays to Prolong the Lifetime of Wireless Sensor Networks. In *Proc. of the 11th ACM MobiCom*, 2005.

[11] M. Grossglauser and D. Tse. Mobility increases the capacity of ad hoc wireless networks. *IEEE/ACM Trans. on Networking*, 10(4):477–486, 2002.

[12] D.B. Johnson, D.A. Maltz, and Y-C. Hu. *The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)*, July 2004. Internet-Draft, draft-ietf-manet-dsr-10.txt. Work in progress.

[13] C.E. Perkins, E.M. Belding-Royer, and S.R. Das. *Ad hoc On-Demand Distance Vector (AODV) Routing*, July 2003. IETF RFC 3561, Network Working Group.

[14] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Trans. on Networking*, 11(1):2–16, 2003.

[15] A. Woo, T. Tong, and D. Culler. Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks. In *Proc. of the 1st ACM SenSys*, 2003.

[16] F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang. A Two-tier Data Dissemination Model for Large Scale Wireless Sensor Networks. In *Proc. of the 8th ACM MobiCom*, 2005.

[17] H.S. Kim, T.F. Abdelzaher, and W.H. Kwon. Minimum Energy Asynchronous Dissemination to Mobile Sinks in Wireless Sensor Networks. In *Proc. of the 1st ACM SenSys*, 2003.

[18] P. Baruah, R. Urgaonkar, and B. Krishnamachari. Learning Enforced Time Domain Routing to Mobile Sinks in Wireless Sensor Fields. In *Proc. of the 1st IEEE EmNets*, 2004.

[19] P. Levis, N. Lee, M. Welsh, and D. Culler. TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications. In *Proc. of the 1st ACM SenSys*, 2003.

[20] J. Polastre, J. Hill, and D. Culler. Versatile Low Power Media Access for Wireless Sensor Networks. In *Proc. of the 2st ACM SenSys*, 2004.

[21] J.-H. Chang and L. Tassiulas. Energy Conserving Routing in Wireless Ad-hoc Networks. In *Proc. of the 19th IEEE INFOCOM*, 2000.

[22] J. Luo. *Mobility in Wireless Networks: Friend or Foe – Network Design and Control in the Age of Mobile Computing*. PhD thesis, School of Computer and Communication Sciences, EPFL, Switzerland, 2006.

[23] M. Demmer and P. Levis. *Tython: A Dynamic Simulation Environment for Sensor Networks*. http://www.tinyos.net/tinyos-1.x/doc/tython/tython.html.

[24] V. Shnayder, M. Hempstead, B. Chen, G.W. Allen, and M. Welsh. Simulating the Power Consumption of Large-Scale Sensor Network Applications. In *Proc. of the 2nd ACM SenSys*, 2004.

[25] R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson, and D. Culler. An Analysis of a Large Scale Habitat Monitoring Application. In *Proc. of the 2nd ACM SenSys*, 2004.