Data Spider: A Resilient Mobile Basestation Protocol for Efficient Data Collection in Wireless Sensor Networks

Onur Soysal, Murat Demirbas

Computer Science & Engineering Dept.,
University at Buffalo, SUNY
{osoysal,demirbas}@cse.buffalo.edu

Abstract. Traditional deployments of wireless sensor networks (WSNs) rely on static basestations to collect data. For applications with highly spatio-temporal and dynamic data generation, such as tracking and detection applications, static basestations suffer from communication bottlenecks and long routes, which cause reliability and lifetime to plummet. To address this problem, we propose a holistic solution where the synergy of the WSN and the mobile basestation improves the reliability and lifetime of data collection. The WSN component of our solution is a lightweight dynamic routing tree maintenance protocol which tracks the location of the basestation to provide an always connected network. The mobile basestation component of our solution complements the dynamic tree reconfiguration protocol by trailing towards the data generation, and hence, reducing the number of hops the data needs to travel to the basestation. While both protocols are simple and lightweight, combined they lead to significant improvements in the reliability and lifetime of data collection. We provide an analytical discussion of our solution along with extensive simulations.

1 Introduction

The objective for deploying a wireless sensor network (WSN) is to collect data from an area for some time interval. Traditionally, a static basestation (SB) is deployed with the WSN, and the WSN nodes relay data over multihops towards the SB, which stores/uploads the data for processing. In order to improve the efficiency (which determines the lifetime) and reliability (which determines the quality) of data collection, most of the research in the literature focus on the relay nodes. Several schemes have been proposed for establishing coordinated sleep-wake-up, aggregation, and routing over relay nodes. On the other hand, relatively little attention is paid to changing the SB model, and investigating holistic solutions to the data collection problem.

The traditional SB model has several handicaps. A primary problem is that the SB constitutes a hotspot for the system. Since the nodes closer to SB are always employed in relaying the entire traffic, those nodes deplete their batteries quickly, putting a cap on the lifetime of the deployment. Another major problem is due to the spatio-temporal nature of the data generation. In several WSN deployments, including environmental monitoring [5], habitat monitoring [16], and especially surveillance systems [1,7], it has been observed that the phenomena of interest are local both in time and space. Fixing the location of the basestation ignores the nature of the data generation and results in long multihop paths for relaying, which leads to a lot of collisions and data losses.

In order to address the drawbacks of the SB model, several work proposed to deploy a mobile basestation (MB) for data collection [2, 8, 11]. The classical "data mule" work [13] proposed to exploit random movement of MBs to opportunistically collect data from a WSN. Here, the nodes buffer their data and upload only when the MB arrives within direct communication. Although this approach eliminates multihop data relaying, the tradeoff is the very high latency, which makes the approach unsuitable for real-time monitoring applications. To fix the latency problem, the mobile element scheduling (MES) work [15] considered the controlled mobility of the MB and studied the problem of planning a path for the MB to visit the nodes before their buffers overflow (which turned out to be an NP-complete problem [9, 15]). MES work assumes that the data-rates in the WSN are known and fixed (constant after initialization), and this is limiting for monitoring applications. Controlled sink mobility in [3] reduces latencies significantly through maintenance of routes to sink location from all nodes. Optimal solution for this model requires preprocessing similar to MES, so the authors in [3] propose a greedy alternative. However, since reactive sink mobility requires flooding of the entire network, the controlled sink mobility work [3] assumes that the sink stays for relatively long durations on small number of predefined sink locations, which limits its ability to address dynamic data generation in an agile manner.

In our previous work, we presented a holistic, network controlled MB algorithm, "data salmon" [6]. Data salmon constructs a backbone spanning tree over the WSN, and constrains both the data relaying and the MB movements to occur on this tree. Data salmon moves the MB greedily to the subtree where most of the traffic originates. [1] In return when the MB moves along one edge of the tree, it inverts the direction of the edge to point to its new location to ensure that the root of the backbone tree is switched to be at the new location of MB. Hence tracking of the MB is achieved with minimum cost. Although it achieves low cost tracking and reduces the average weighted relay distance of data, the data salmon has some shortcomings. The hotspot problem is not addressed: since data salmon uses a static backbone tree, the center of the static backbone tree still relays a significant amount of traffic and is a potential hotspot. Moreover, a static backbone tree implies that a message-loss during the handoff of MB from one node on the tree to the next leads to a permanent partitioning. [2]

Our contributions. We present **data spider** a holistic solution where the synergy of the WSN and the MB improves both the reliability and lifetime of data collection. The WSN component of our solution is a very lightweight

[1] We showed that this greedy strategy is optimal, under the constraints of limiting all the data relaying to occur on the static backbone tree.

[2] Requiring acknowledgment messages alleviates the problem, but also increases the overhead of the protocol significantly.

dynamic routing tree maintenance protocol which tracks the MB to provide an always connected network. This tree is updated locally and efficiently by the movements of the MB. The visual imagery is that of a spider (corresponding to the MB) re-weaving/repairing its web (corresponding to the tree) as it moves. To complete the feedback loop, the spider relies on its web to detect interesting phenomena (data generation) to follow. By trailing towards the data generation, the MB component of our solution reduces the number of hops the data needs to travel to the basestation. Since the data spider uses a dynamically reconfigured tree to route traffic, it avoids the hotspot problem of data salmon, which used a static backbone tree for routing. As a result, data spider extends the lifetime of the deployment by several folds over the data salmon. Due to its dynamically reconfigured tree, data spider is also resilient against message-losses.

We present our dynamic tree reconfiguration protocol, DTR (read *detour*), in Section 3. The DTR philosophy is to update the tree at where it counts, that is, where the most recent action is. Therefore, instead of trying to maintain a distance-sensitive tree for the entire network (which is clearly a non-local task), we maintain a temporally-sensitive tree by reconfiguring the tree only at the immediate locality of the MB. Since we restrict the tree reconfiguration only to the singlehop of the MB, the maintenance cost of the tree is very low. Yet this does not lead to long and inefficient paths for data relaying to MB: since the MB follows the data generation closely, the effective length of data-forwarding paths is only a couple of hops. (The lenghts of non-data-forwarding paths are irrelevant for the data collection problem.) We give the correctness proof of tree reconfiguraton at DTR in Section 3.2.

To investigate the requirements for proper handoffs by the MB, we formulate the *handoff connectivity* property in Section 3.3. Handoff connectivity, intuitively, captures the notion of having no holes in the network. We note that our data spider waives the handoff connectivity requirement in practice. In Section 4 we present a simple yet very effective algorithm—trail-flow algorithm—for the MB, that avoids bad handoffs. In the trail-flow algorithm, the MB follows the edges where most data is flowing to itself, instead of going directly to the source of the data (which we dub as the follow-source approach). Our simulation results show that follow-source leads to several incorrect handoffs whereas trail-flow still functions correctly in the same density/network.

We give simulation results to investigate the scalability and efficiency of data spider, and compare it with data salmon and the SB approach in Section 5. Our simulator uses realistic lossy channel models and provides a high-fidelity energy calculation by using BMAC [12] as the model for the MAC layer communications. [3] Our simulation results show that data spider outperforms data salmon and the SB approaches consistently, and leads to significant improvements in the reliability and lifetime of data collection. Although we focus on one mobile region of interest (ROI) and on one MB for most of the paper, we note that data spider

[3] Since our simulator is parametrized extensively it is suitable for modeling and investigating other MB algorithms easily. Our simulator is available at `http://www.cse.buffalo.edu/ubicomp/dataSpider/`

extends readily to allow several MBs to share the same network to track multiple ROIs. Our simulation results with multiple MBs collecting data from multiple ROIs are very promising; despite the lack of explicit coordination between the MBs, these simulations show an emergent cooperation and division of labor among the MBs leading to improved performance.

2 Model

We consider a dense, connected, multihop WSN. The sensor nodes are static after the initial deployment. We assume that the data generation has spatio-temporal correlation but is otherwise dynamic/unpredictable. This model captures the data generation in event detection, tracking, and surveillance applications. Our implementation of the data generation uses a circular region of interest (ROI). We allow only the sensors in this ROI to generate data. The ROI moves around in the network nondeterministically to simulate the behavior of mobile events. This ROI scheme generates dynamic data that is challenging for precomputed basestation mobility as in [3, 9].

We account for the message transmission costs of the sensor nodes as well as the reception costs and idle listening costs at the nodes. We assume CSMA with BMAC low-power-listening [12] for the MAC layer and use the associated energy model in [12] to calculate the energy usage at each node. Since our DTR and MB protocols are simple, we ignore the energy cost of the computation. We assume the MB is capable of locating itself and traveling to target locations. In our model, we have not included the energy required for moving the MB. [4]

3 Dynamic Tree Reconfiguration

Here, we first present the DTR algorithm. We give the correctness proof of DTR in Section 3.2 and present the handoff connectivity requirements for DTR in Section 3.3. Finally, we present extensions to the basic DTR in Section 3.4.

3.1 DTR Algorithm

To maintain always-on connectivity to the MB, the network should continuously track it and update the existing routing paths to point to its new location. Trying to maintain a distance-sensitive tracking structure (e.g., maintaining a shortest path tree rooted at the MB) would be beneficial since it would reduce the number of hops data need to be relayed towards the MB. However, this is inherently a non-local and costly task as it requires frequent multihop broadcasts.

[4] The reason behind our willingness to generously tradeoff the energy required for relocating the MB with the energy gain in data collection is that it is much easier to replenish and maintain the batteries of one MB than those of the sensor nodes in the entire network. We assume that the MB is recharged periodically or is equipped with energy harvesting capabilities such as solar panels.

Since energy-efficiency is of utmost importance for elongating the lifetime, in our dynamic tree configuration protocol, DTR, we take an alternative approach. To keep the maintenance cost of the tree very low, we confine DTR to reconfigure the tree only at the immediate locality (singlehop) of the MB. To ensure that DTR does not beget long and inefficient paths for data relaying to MB, we rely on the MB algorithm. In our simulation results in Section 5.2, we show that since the MB's trail-flow algorithm follows the data generation closely, the effective length of data relaying paths is only a couple of hops.

DTR starts with a spanning tree rooted at the MB. This could be established by constructing an initial tree using flooding and keeping the MB static. The root node of this initial tree is called the *anchor* node, which is also the closest node to the MB. As it relocates in the network, MB chooses the anchor node to be the closest node to itself and makes periodic broadcasts to declare the anchor node to all nodes in its singlehop range. Nodes that receive the anchor broadcast update their parents (next pointers) to point to the new anchor node. At any time there is a unique anchor node in the network, which is maintained to be the closest node to the MB.

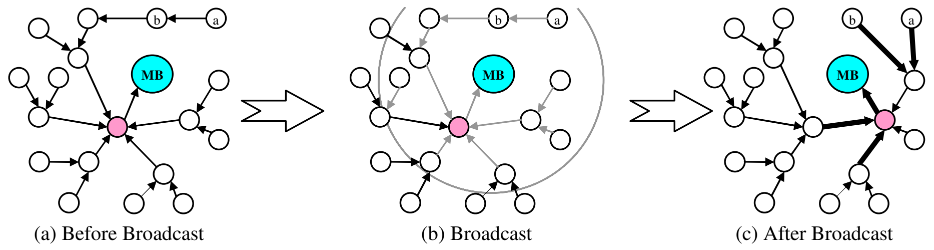


Fig. 1. Demonstration of DTR as MB moves from one anchor to another. The touched edges are gray edges in (b) and actual changes are bold edges in (c).

We present DTR in Algorithm 1. Only the nodes that receive the anchor broadcast execute an action and update their next pointers. The anchor broadcasts are local to the singlehop of the MB and they are not relayed to multiple hops. Figure 1 depicts an example of DTR execution.

Algorithm 1 DTR Algorithm

```
1: Wait for the anchor message
2: if anchor == self then
3:     next ← MB
4: else if anchor ∈ Neighbors then
5:     next ← anchor
6: end if
```

Dynamic convoy tree work [17] adresses a relevant dynamic tree reconfiguration problem in the context of target tracking. Dynamic convoy tree maintains a monitoring tree to cover a mobile ROI. The root of this monitoring tree controls expansion and contraction of the tree and when needed decides on the relocation of the root to another node based on the information it collects from the entire tree. Our advantage in DTR is the cooperation of the MB for relocating the root of the tree to an optimal location using local singlehop updates, whereas the convoy tree needs to deal with the tree reconfiguration problem by using multihop update messages.

3.2 Correctness

We call the operation with which the MB changes the anchor node from one node to another as *handoff*. We call a handoff a *proper handoff* iff (1) both the old and new anchor nodes receive the MB's anchor broadcast, and (2) both the old and new anchor nodes can reliably communicate with each other. Finally, we call a network *routing connected* iff the *next* links of nodes form a spanning routing tree rooted at MB.

Theorem 1. *If all handoffs are proper, an iteration of Algorithm 1 starting from a routing connected network results in another routing connected network.*

Proof. Consider tree reconfiguration on a graph $G = (V, E)$ where $u, v \in V$ correspond to the nodes and $e = (u, v) \in E$ correspond to the reliable communication links between the nodes. We use r to denote the old anchor and r' to denote the new anchor. In the base case, when there is no handoff, $r' = r$, and the theorem holds vacuously. We next consider the case where $r' \neq r$.

The iteration of Algorithm 1 entails an anchor broadcast received by a set of nodes $R \subset V$. Let $S \subseteq R$ be the set of nodes that actually change their *next* links as a result of executing Algorithm 1. Since proper handoffs are assumed, $\{r, r'\} \subseteq S$. Algorithm 1 dictates that all nodes in S points to r' (with the exception of r' which points to the MB) after the update. That is, the *next* links of nodes in S form a routing tree rooted at r'.

Let $T(r)$ be a spanning routing tree of G rooted at node r, and F_S be the forest obtained by removing the *next* links of nodes in S from $T(r)$. Since $r \in S$, each tree in F_S is rooted at a node in $s \in S$. By definition, none of the edges in any tree $T_s \in F_S$ is changed. Since *next* links in S forms a routing tree rooted at r', *next* links in F_S and S form a spanning routing tree rooted at r'.

While message losses are common in WSN environments, most message losses do not affect the correctness of DTR (Theorem 1), as the definition of proper handoffs only require reliable message delivery between the MB and the old and new anchors. For the remaining nodes, message loss is only a nuisance, rather than constituting a correctness problem. Message losses at these nodes may result only in degraded performance, since their path is not updated to point to the new anchor in the most direct/shortest manner. But, since the previous routes point to the old anchor, which points to the new anchor, due to Theorem 1 the network is still routing-connected.

The routing-connected network property is violated only when the old or new anchor miss an anchor broadcast. DTR deals with this problem in two timescales: short and long terms. In the short term the impact of message losses are reduced through message redundancy. Increasing the anchor broadcast frequency at the

MB improves the chances that all neighbors receive the information about the new anchor node. When this scheme is insufficient, there may be partitions in the network due to improper handoffs. In the long term, since the MB is mobile, MB is very likely to move over the partitioned regions eventually. This will, in turn, fix the problem and enable the buffered packets to be relayed to the MB.

3.3 Handoff Connectivity

The correctness of DTR depends on the success of handoffs, which is in turn imposed by the geometry and topology of the network. Here, we focus on planar deployments and capture these required geometric and topological properties.

In data spider, MB invariantly maintains its closest node as the anchor node. A useful abstraction for capturing this property is the Voronoi diagram of WSN nodes. When the MB is in one of the Voronoi cells, its closest node, by definition, is the WSN node corresponding to that Voronoi cell. Thus, as long as MB stays in that Voronoi cell, the anchor node is unchanged.

With this anchor node definition, we identify the requirements for having proper handoffs as follows. Let P denote a point in the deployment area and V_P be the set of nodes which are closest to P. So, if P falls inside a Voronoi cell, then V_P consists of a single node, the WSN node corresponding to that Voronoi cell. If P falls on a Voronoi cell boundary, then V_P consists of the neighboring (i.e., adjacent) nodes for this Voronoi cell boundary.

We call a WSN deployment **handoff connected** when all points P in deployment region satisfy:

1. For all nodes $n \in V_P$, n can reliably communicate with a node placed at P.
2. For all nodes $n, m \in V_P$, n and m can reliably communicate with each other.

In other words, in a handoff connected network (1) the MB sitting on a Voronoi cell boundary can communicate with the nodes in the adjacent Voronoi cells, and (2) any pair of Voronoi neighbors can communicate with each other.

The above handoff connectivity definition is valid when the updates of the MB are continuous. Since we use discrete/periodic anchor broadcasts, we extend this definition for our model. Let $\lambda = v_{BS} * T_{update}$ be the maximum distance the MB can travel between two location updates. We now require the anchor node to be able to receive messages from the MB when it is at most λ away from the Voronoi cell. Moreover, for proper handoff, any cell that falls to this region should be in communication range. Figure 2 demonstrates this requirement.

To generalize the handoff region we extend the set of nearest nodes V_P. V_P^λ to be the set of nodes which are at most $\lambda + d_{min}$ away from point P where d_{min} is the minimum distance to any node in network from P. Thus, using V_P^λ we generalize *handoff connectivity* as follows:

A WSN deployment is said to be **λ-general handoff connected** when all points P satisfy:

1. For all nodes $n \in V_P^\lambda$, n can reliably communicate with a node place at P.
2. For all nodes $n, m \in V_P^\lambda$, n and m can reliably communicate with each other.

3.4 Extensions to DTR

Handoff connectivity addresses only the immediate neighborhood of the anchor node. Broadcasts on the other hand can be made stronger with better transmitters on the MB so WSN nodes can receive broadcasts from non-neighboring Voronoi cells. These receptions can be utilized to improve the performance of DTR as follows. For this operation, nodes depend on neighborhood information about their neighbors. That is, nodes share neighborhood information with their neighbors, so that they can create two-hop routes to the anchor node when singlehop routes are not possible. If the anchor is not an immediate neighbor of the node, the node chooses its neighbor which is an immediate neighbor of the anchor. In case there are multiple neighbors satisfying this condition, the closest one to the anchor is chosen as the next node. As long as the chosen intermediate nodes also received the anchor broadcast this operation extends the handoff connectivity. We call this operation *indirect handoff*. We show neighbor nodes where only indirect handoff is possible with green(light) edges in Figure 2. Non-anchor nodes also benefit from our indirect handoff extension, as is the case for nodes a and b in Figure 1.

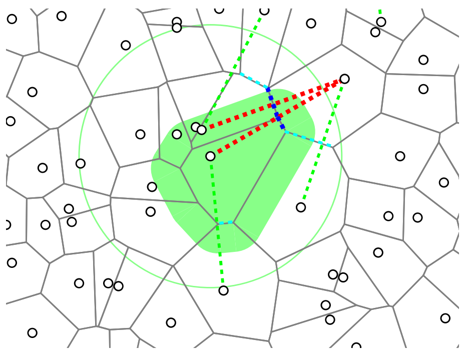


Fig. 2. Shaded region shows possible locations of MB at the next update, starting from center. Circle denotes the reliable communication range. Green(light) dashed lines correspond to neighbors where direct handoff is not possible. Red(dark) dashed lines correspond to neighbors where no handoff is possible.

4 MB Algorithm

The MB algorithm (given in Algorithm 2) synergizes with the DTR algorithm to achieve efficient data collection. MB ensures two things:

(1) The MB broadcasts an anchor message announcing the closest node to itself periodically. This enables DTR to track MB correctly and update the tree to be rooted at the MB. In order to detect and announce the closest node to itself, MB should know its location as well as the locations of nodes in the network. This is achievable by equipping the MB with a GPS and the coordinates of the WSN nodes. Having a GPS on the MB is relatively cheap, and the MB can also utilize its GPS to locate and collect the nodes after deployment.

Algorithm 2 MB Algorithm

```
1: loop
2:     listen and update RecentPackets
3:     if count(RecentPackets) $> 0$ then
4:         target $\leftarrow$ getTarget()
5:     else
6:         target $\leftarrow$ getRandomTarget()
7:     end if
8:     navigate to target
9:     anchor $\leftarrow$ closestNodeTo(position)
10:    broadcast anchor message
11: end loop
```

(2) The MB relocates to follow data generation in a best-effort manner. This relocation reduces the length of data relaying paths in DTR to be a couple of hops, improving both the reliability and the lifetime. To track the data generation, MB utilizes the recent data packets that DTR routes to itself to decide where to move to next. MB defaults to a random walk when there are no packets, since this might indicate a disconnection of the network. Random walk may help the MB to repair the partitioning and re-establish a connected network where DTR can start delivering the data generated to the MB. Otherwise, MB uses the *getTarget()* function to decide how to relocate based on the recently received packets. We propose two heuristics for this function:

trailSource. Here, the MB inspects the source field of the data packets and sets the relocation target to be the source of the packet generation (median of the source locations). Although it seems like a reasonable approach, we show in Section 5 that when the network is not regular (has holes in it) trailSource leads to many improper handoffs and suffers severe performance penalties.

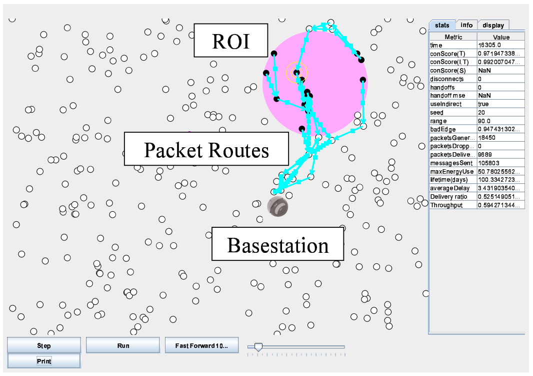
trailFlow. Here, the MB tries to go to the center of packet flow. In contrast to trailSource that calculates the center of data generation, trailFlow calculates the center of data forwarding from the singlehop neighbors of the MB. Since packet forwarding is done over reliable edges, trailFlow directs the MB to avoid the holes in the network implicitly (as a side benefit), so even in irregular and sparse networks trailFlow ensures successful handoffs. Our simulation results in Section 5 show that trailFlow consistently performs the best compared to the other heuristics.

5 Simulation

5.1 Setup

Simulator. We built our simulator on top of the JProwler simulator [14] and implemented support for mobility for JProwler. Our simulator is parametrized extensively, so it is suitable for modeling and investigating other MB algorithms quickly. Figure 3 shows a screenshot of our simulator. Our simulator and details about our simulator implementation are available at `http://www.cse.buffalo.edu/ubicomp/dataSpider/`.

Simulation setup. We set the simulation area as a 160m by 120m rectangular region with 300 mica2 nodes. We constrain the MB to this region and assume that there are no significant obstacles to obstruct mobility within the region. We model the data generation activity in the environment with a moving disc to denote the ROI. The region of interest is a circle with 20m radius. All WSN nodes covered by this disc generate data with a predetermined rate. The nodes then try to forward this data to the MB if they have a valid *next* link. A node buffers data if the channel is busy, or if it does not have a valid *next* link—which may happen after an improper handoff. We leverage on work in [10] to generate realistic human/animal like mobility patterns for the ROI.



Metric	Value
time	16305.0
conScore(T)	0.971947338...
conScore(I,T)	0.992007047...
conScore(S)	NaN
disconnects	0
handoffs	0
handoff mse	NaN
useIndirect	true
seed	20
range	90.0
badEdge	0.947431302...
packetsGener...	18450
packetsDropp...	0
packetsDelive...	9689
messagesSent	105803
maxEnergyUse	50.78025562...
lifetime(days)	100.3342723...
averageDelay	3.431903540...
Delivery ratio	0.525149051...
Throughput	0.594271344...

Fig. 3. A screenshot of the simulator. Dark circles are the data generating nodes, and shaded (pink) circle is the ROI. MB is indicated by the tiny roomba picture and arrows show the packet routes.

In order to address energy efficiency questions we keep track of energy use in our simulation. Our simulator uses CSMA with BMAC low-power-listening [12] for the MAC layer and the associated energy model to calculate the energy used in each node. In our simulation we obtain fine grain information about packet arrivals and noise and replace the approximate values used in [12] with these values to better capture the energy use in each sensor node. Table 1 summarizes the parameters used for the energy efficiency calculations.

Parameter	**Value**
Radio sampling interval	0.1s
Energy cost of a packet transmission	7.62mJ
Energy cost of a packet reception	3.18mJ
Energy cost of LPL for one second	0.263mJ
Battery Capacity	27000J

Table 1. Parameters used for energy efficiency calculations

We ran each set of simulations for 72 simulation hours. Each simulation includes an initial neighborhood discovery and initial flooding phase. Neighborhood discovery phase reduces the disconnections and message losses as reliable links are identified and each node discovers its neighbors. This neighborhood information is later utilized in performing indirect handoffs. We **do include** the communication in this phase in our energy cost figures as well.

Protocols we compare with. We are primarily interested in evaluating the data spider system which consists of DTR and the MB algorithms, trailSource and trailFlow, described in Sections 3 and 4. For comparison, we also consider three other protocols, namely, *static*, *random*, and *salmon*.

In the static protocol, the basestation is static and is located in the center of the network. The data is routed to the based using a convergecast tree rooted at the SB. As we discussed in the Introduction this scheme is prone to hotspots around the SB, and also results in long multihop paths for data relaying.

The random protocol is similar to trailSource and trailFlow in that it also uses the DTR protocol to reconfigure the data collection tree as the MB relocates. However, as for the relocation algorithm, instead of trying to follow the data generation, the random protocol prescribes relocating the MB to a random location all the time. While this protocol avoids the hotspot issue (since it uses an MB and DTR), it is prone to long multihop paths for data relaying as it does not follow the data generation.

Salmon protocol uses the same MB algorithm we used in our previous work, data salmon [6]. Salmon does not use DTR and constrains the relocation of the MB to occur only along the edges of the existing tree. In other words, the existing tree is not modified, except for the relocation of the root of the tree from one node to one of the neighboring nodes (which is achieved by flipping the direction of the edge between these two nodes). In this scheme, the MB chooses the neighbor that forwards the majority of the traffic to relocate to. As our simulation results exhibit, this scheme has problems with reliability (since only one edge is modified, this constitutes a risk of single point of failure) and cannot follow the data generation successfully (since the MB relocation is restricted to the existing tree structure, MB needs take long detours when the ROI leaves the current subtree for another subtree).

Metrics. We concentrate on three metrics to measure performance of the system. The latency metric measures the average delay in packet deliveries, from their generation time to their arrival to MB. The second metric, packet delivery rate, is the ratio of the delivered packets to MB versus the number of packets generated. The final metric is the estimated lifetime of the network. We define the lifetime to be the time passed until the first node failure due to battery depletion in the network. By utilizing the fine-grained energy-use information from our simulation and the total energy stored in standard AA batteries, we arrive to our estimated lifetime figures.

5.2 Results

We present our simulation results under the following categories.

Node density. We first investigate the effect of node density on the performance. As the number of nodes increase, since the distance between anchor nodes would be decreasing, we expect better connectivity of the network and reduced number of improper handoffs. Increased density also corresponds to increased data rates and more contention reducing the lifetime of the network. Figure 4 presents this axis of the investigation. We observe very high latencies when node density is low. This is due to frequent disconnections. Packets are buffered when handoffs can not be completed successfully and they are later retrieved on an opportunistic basis, but this results in high average latencies. Data spider heuristics *trailFlow* and *trailSource* consistently outperform other protocols with respect to packet deliveries and network lifetime. An interesting result of this experiment is to show that even random mobility leads to better delivery ratios than the static when the density is critically low. Random mobility leads to worse delivery ratios when the density increases, yet it still leads to longer lifetimes than static (recall that random still uses DTR for data collection).

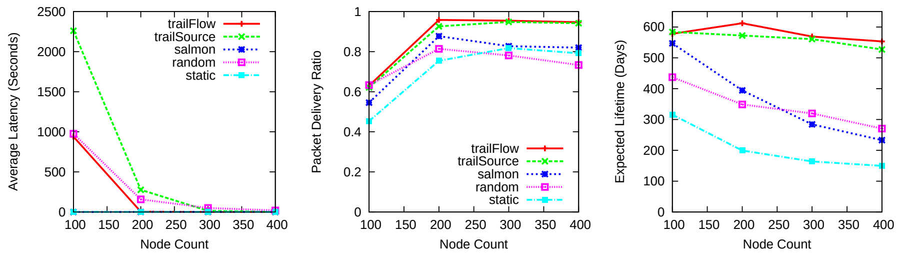


Fig. 4. Effect of number of nodes on performance

Indirect handoff. Here we quantify the performance improvement due the indirect handoff extension. Our experiments in Figure 5 show that indirect handoff provides better average latencies, and up-to 5% improvement in packet delivery rates. The benefit of indirect handoff is most significant in expected lifetime which is improved by 20% for low network density.

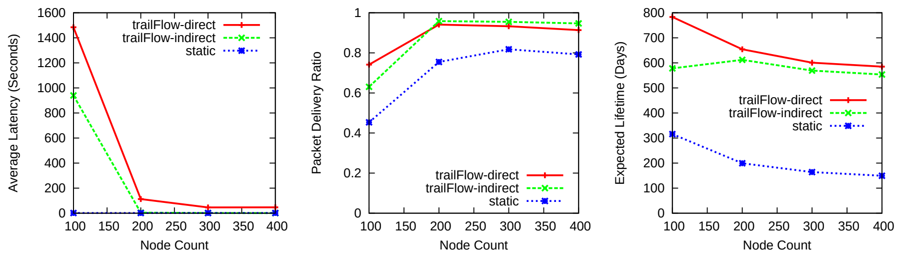


Fig. 5. Effect of indirect handoff on performance

Speed of region of interest (ROI). The ability to track ROI is a significant advantage for data spider, but the performance of tracking is affected by the speed of ROI. In our experiments depicted in Figure 6 we investigate the effect of speed of ROI to the performance. Since we use a fixed speed for MB, increasing the speed of ROI makes tracking the data more difficult. As expected static and random heuristics are not effected by the ROI speed. We observe significant increase in average delay in *trailSource* heuristic. This increase is related to increased number of bad handoffs, which leads to partitions of network. *trailFlow* avoids this problem as packets follow the network topology and the MB follows the packets. Even with increased ROI speed, data spider algorithm improves the lifetime of network up to 3 times over SB.

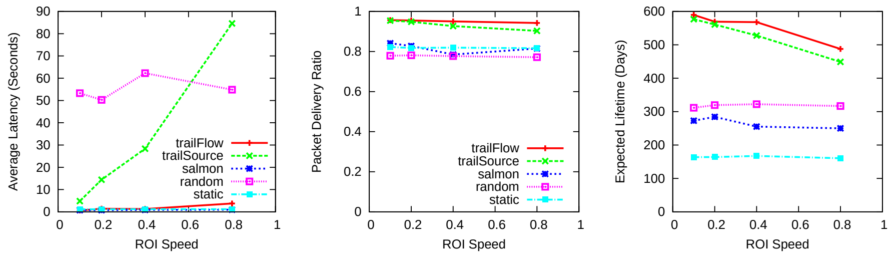


Fig. 6. Effect of speed of region of interest on performance

Number of ROI. We next consider the effects of increasing number of ROIs on the performance. As these ROIs move independently from each other, the optimal location of MB would vary significantly and the static MB starts to become a better alternative. Our simulation results are shown in Figure 7. We observe the effect of disconnections in *trailSource* heuristic in this experiment as well. The difference between data delivery rates decrease as data spider heuristics can not follow all the ROIs at the same time. Lifetime of the network is also inversely affected as the MB is constrained to a smaller region trying to follow all ROIs simultaneously. With 4 ROIs, the performance of data spider is similar to random MB in terms of network lifetime, which is still more than 100% improvement over the SB.

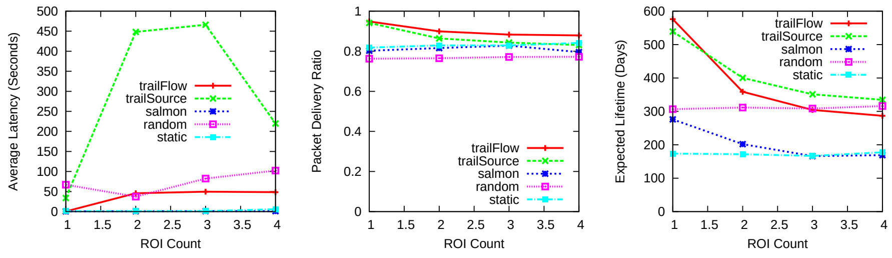


Fig. 7. Effect of number of region of interests on performance

Multiple MBs. Figure 7 showed that increasing the number of ROIs reduced the ability of data spider to track them. Here we show how the increased number of ROIs are better handled with multiple MBs. We test the performance of data spider with multiple MBs in Figure 8. As we mentioned in the Introduction, data spider extends readily to allow multiple MBs to share the same network without any need to change the DTR or MB algorithms. In this experiment neither the network nor the MBs are aware of the multiple MBs. However, we still observe an emergent cooperation and division of labor leading to improved performance. MBs partition the network since each node only has one next node, moreover these partitions dynamically change over time due to MB broadcasts. Even if all MBs converge to same anchor, the competition for data allows MBs to diverge and cover different ROIs. We obtained these very promising results with data spider despite lack of explicit coordination. An interesting research question is how to coordinate MBs in a cooperative manner to improve performance even further. Recent studies focus on this direction [4].

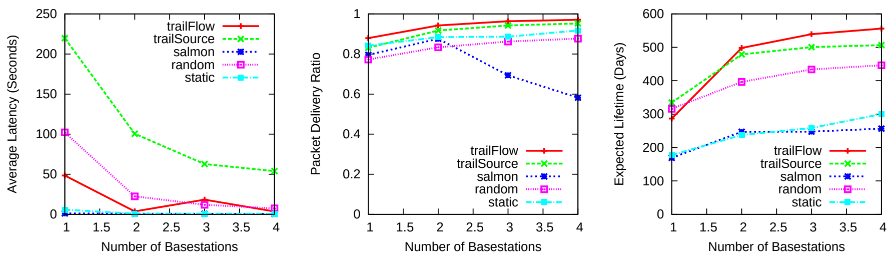


Fig. 8. Effect of number of MBs on performance with 4 independent ROIs.

6 Concluding Remarks

We presented an efficient holistic MB-based data collection system, data spider, which consists of a dynamic tree reconfiguration protocol and an MB protocol. While both protocols are simple and lightweight, combined they lead to significant improvements in the reliability and lifetime of data collection, especially for monitoring applications with highly spatiotemporal data generation. We provided extensive simulation results evaluating the latency, cost, and network lifetime metrics of the data spider under a wide number of varying parameters. We also analyzed the handoff connectivity requirements needed for performing a proper handoff of the MB.

Although we focused on the data collection problem, our data spider framework readily applies also to the pursuer-evader tracking problem by treating the ROI as the evader and the MB as the pursuer. Our experiments showed that the trail-flow algorithm for the MB manages to implicitly route the MB around the holes, a desirable property for pursuer-evader tracking. Our experiments also showed that, in the data spider system, multiple MBs coexist nicely on the same network to trail multiple ROIs without any explicit coordination or cooperation. In future work we will investigate coordination and cooperation mechanisms of multiple MBs for more efficient pursuer-evader tracking.

References

1. A. Arora and et. al. Exscal: Elements of an extreme scale wireless sensor network. *RTCSA*, 2005.
2. A. Azad and A. Chockalingam. Mobile base stations placement and energy aware routing in wireless sensor networks. In *WCNC*, volume 1, pages 264–269, April 2006.
3. S. Basagni, A. Carosi, E. Melachrinoudis, C. Petrioli, and Z. M. Wang. Controlled sink mobility for prolonging wireless sensor networks lifetime. *Wirel. Netw.*, 14(6):831–858, 2008.
4. S. Basagni, A. Carosi, and C. Petrioli. Heuristics for lifetime maximization in wireless sensor networks with multiple mobile sinks. In *Communications, 2009. ICC '09. IEEE International Conference on*, pages 1–6, June 2009.
5. M. Batalin and et.al. Call and response: experiments in sampling the environment. In *SenSys '04*, pages 25–38, 2004.
6. M. Demirbas, O. Soysal, and A. S. Tosun. Data salmon: A greedy mobile basestation protocol for efficient data collection in wireless sensor networks. *DCOSS*, pages 267–280, 2007.
7. A. A. et.al. A line in the sand: A wireless sensor network for target detection, classification, and tracking. *Computer Networks (Elsevier)*, 46(5):605–634, 2004.
8. S. Gandham, M. Dawande, R. Prakash, and S. Venkatesan. Energy efficient schemes for wireless sensor networks with multiple mobile base stations. In *GLOBECOM*, volume 1, pages 377–381 Vol.1, Dec. 2003.
9. Y. Gu, D. Bozdag, E. Ekici, F. Ozguner, and C. Lee. Partitioning based mobile element scheduling in wireless sensor networks. *SECON*, pages 386–395, 2005.
10. W. jen Hsu, T. Spyropoulos, K. Psounis, and A. Helmy. Modeling time-variant user mobility in wireless mobile networks. In *INFOCOM*, pages 758–766, 2007.
11. J. Luo and J.-P. Hubaux. Joint mobility and routing for lifetime elongation in wireless sensor networks. In *INFOCOM*, volume 3, pages 1735–1746, 2005.
12. J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *SenSys*, pages 95–107, 2004.
13. R. C. Shah, S. Roy, S. Jain, and W. Brunette. Data mules: modeling a three-tier architecture for sparse sensor networks. *WSNPA*, pages 30–41, 2003.
14. G. Simon, P. Volgyesi, M. Maroti, and A. Ledeczi. Simulation-based optimization of communication protocols for large-scale wireless sensor networks. *IEEE Aerospace Conference*, pages 255–267, March 2003.
15. A. Somasundara, A. Ramamoorthy, and M. Srivastava. Mobile element scheduling for efficient data collection in wireless sensor networks with dynamic deadlines. In *RTSS*, pages 296–305, 2004.
16. R. Szewczyk, A. Mainwaring, J. Polastre, and D. Culler. An analysis of a large scale habitat monitoring application. *SenSys*, 2004.
17. W. Zhang and G. Cao. Dctc: Dynamic convoy tree-based collaboration for target tracking in sensor networks. *IEEE Transactions on Wireless Communication*, 3(5):1689–1701, 2004.