Improving Non-Stationary Data Retrieval in Wireless Sensor Networks

Andrew LeBeau Justin Fields Ryan Lavering
Diana Franklin John Seng

Department of Computer Science
Cal Poly State University
San Luis Obispo, CA 93407

Abstract

Wireless sensor networks provide an effective tool for acquiring sensor measurements across a large area. Devices in a sensor field can automatically configure themselves into a working network, ready to collect data. The challenge is to collect that data in a timely and power-efficient manner.

In some environments, a data collection device (sink) may need to travel periodically through the field to gather data because keeping a local storage system is impractical. The goal is for this moving sink to gather the most data in the shortest amount of time, using the least amount of inter-node communication. Network reconfigure time, memory limitations, and brief radio contact time challenge this goal.

In this work, we explore three aspects of data retrieval. First, we explore with what nodes the sink should communicate. We present a selection technique for picking target nodes to increase data retrieval rates. We then optimize data collection through message forwarding and exclusion. These two techniques improve data retrieval by 35% on average. Finally, we consider the effect and efficiency of various sink paths.

1. Introduction

Wireless sensor networks are an effective means of collecting data across a large area. Each node of a sensor network can be placed in locations that are relatively far apart, and the information each node acquires can be relayed through other nodes in the network and collected in a central location. This type of data collection enables many useful applications ranging from monitoring ecological habitats[3] to monitoring the seismic activity over a large area of land[2].

In this paper, we study algorithms targeting data retrieval from a wireless sensor network where data collection cannot be performed by routing data to a stationary location. The data must be retrieved by a mobile data collector (sink) which passes through the field of sensor nodes. For example, a vehicle (e.g. car or airplane) may travel through a sensor field to retrieve current measurements from each of the nodes. Often, the vehicle will be moving too quickly to retrieve all the data in a single pass. Additionally, each node's memory may be too small to store data from all its neighbors. Data must either be sent directly to the data sink or routed to a node which is in communication range with the data sink. In this research, we study algorithms to collect data from as many nodes as possible.

The primary challenge of this type of data collection is that the *target node*, the current node in communication with the data sink, is in communication contact with the data sink only for a finite period of time. Each time the target node changes, the other nodes must reconfigure themselves to send to the new target node, and some must retransmit their data that had not yet reached the sink. Unfortunately, this costs time and power. We attack this problem from several angles - by reducing the number of times the target changes and by optimizing the case in which it changes. The selection of the target node is crucial because each time the target node gets out of range, extra messages and network reconfiguration are required. We present two optimizations for when the target node changes. First, we reduce messages (and thus power) by removing some of the redundant messages from the system. Once this is done, however, the original messages need to be re-routed to the new target node. Secondly, we present a form of *data forwarding* which routes data towards the target node, in the presence of target node changes. By utilizing this data forwarding, we are able to increase the amount of data collected from 31% to 66% on average.

Finally, we show that our algorithm works well independently of the path taken. When normalized for total time spent in the field, non-linear paths and linear paths all performed within 8% of each other on average.

This paper is organized as follows: Section 2 describes related research and studies, whereas Section 3 describes our work. The simulation methodology, assumptions, and tools are presented in Section 4, followed by results in Section 5. Section 6 describes future work, and we conclude in Section 7.

2. Related Work

Within the context of sensor networks, there are several interesting problems in transferring information from a field of sensors to a data sink. We focus on how a moving sink collects data and how to maximize data collection given limited radio contact.

TAG [5] creates a language and library for aggregating data from many motes to a single, stationary sink. It reduces processing and storage by calculating results as the messages propagate through the system. [4] takes a step back and analyzes the problem of data aggregation in terms of energy effi-

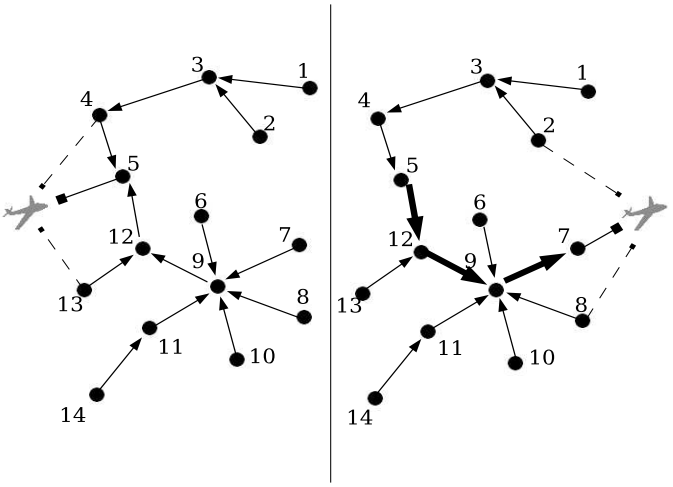

Figure 1. The sensor network as a plane gathers data.

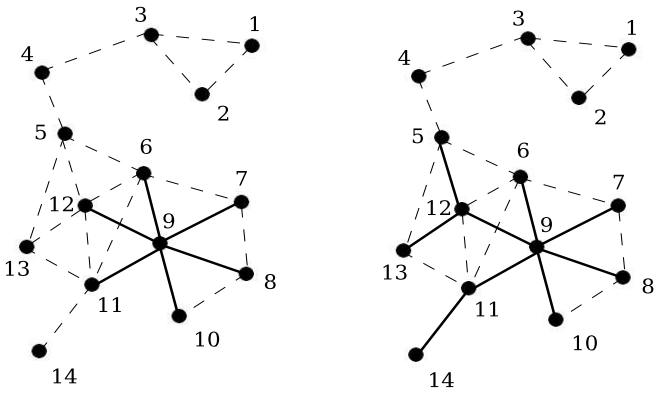

Figure 2. Two steps in linking setup phase. The node with the most neighbors is the start node. As neighboring nodes are added, the nodes with the fewest edges are added first.

ciency and delays. They find that although optimality is NP-Hard, special polynomial time cases also exist.

Other groups have looked at quickly and efficiently creating networks between source nodes and sink nodes. [6] proposes a cost field that, with few messages, creates a minimum cost forwarding route for each node to a stationary sink. In addition, it requires no centralized information to maintain its routing.

[1] has created a system to allow nodes to pass data with both efficiency and power-management in relation to the path they choose in order to get the data to the sink.

Finally, there has been some work on collecting data from a network to a non-stationary sink. Zebranet [3] provides nodes with enough memory to store the data from neighboring nodes so that the communication occurs before collection, not during collection. We are assuming simpler nodes that do not have this much memory.

3. Non-Stationary Data Retrieval

In this work, we focus on optimizing data collection from a field of sensor nodes via a non-stationary sink node. The sink node may be an airplane or car traveling through a field of fixed-location sensors. In any given application, the goal is to obtain the data from as many of the sensor nodes as possible. A sample scenario is depicted in Figure 1. The figure shows an airplane as it approaches (a) and travels through (b) a sensor network. First, the airplane must choose a primary node for initial communication, a *target node*. Data is routed toward this target node. The target node changes as the airplane comes in and out of communication range with different nodes.

Because the sink node is moving, any one of the sensors may have radio contact with the sink for a brief period of time, and some nodes do not make direct contact with the sink at all. This limits the amount of information that can be collected from the nodes. Collecting all the data is achieved by routing data through other sensor nodes.

In an ideal situation, once the airplane chooses a target node, the nodes would link themselves such that they could pass messages to that target node in the fewest hops possible. Unfortunately, a sink traveling at a high speed will leave the area before such an optimal network is set up, let alone before it has time to receive all messages. We use a linking algorithm for our studies, whereby the sensors form an initial network of bidirectional links at setup time (long before data collection), and only the direction of the links is changed depending on the target node at a given time.

The linking setup algorithm is depicted in Figure 2. Dashed lines represent all possible links, whereas solid lines represent links selected by the algorithm. The links are created by first finding the node that is in communication range of the most nodes (node 9 in the example). In order to load-balance messages, the algorithm adds nodes that have the least solid-line links (as opposed to potential links) until all nodes are connected. In the above example, it would link all of node 9's neighbors, and then look for the next set connected to those nodes.

This linking algorithm leads to trees with low depth (assuming the first node selected is the root). This reduces the penalty for new target node selection because the data will not need to be redirected as far. By starting in a high-density area of the field and branching out, it is likely that the next level of nodes will not have many links. By continuing to branch out based on fewest links, it is likely that the nodes with many potential links will not be able to link to all their neighbors because the surrounding nodes will already be linked.

Note that this is a tradeoff. We are focusing on speed while sacrificing some power and memory pressure on the first node chosen.

In the end, each node will only receive messages from or send messages to its links; all other messages will be excluded. From this setup, we explore several algorithms for how to choose the target node, how to most efficiently pass data to the sink, and what path the sink should take.

When the sink enters the field and selects the first target node, the bidirectional links are made unidirectional, all leading to the target node. Messages are put in a send queue for each node that sends to their parent node. Each node forwards one message every time the sink sends a message to the target node. Messages are forwarded as long as the sink is within range of the target node. Once the sink is no longer within range, a new target is selected.

This process of target selection and data retrieval is repeated until the sink leaves the sensor field. After the sink has not been in the field for an extended period of time, each node clears its message sending queue.

4. Methodology

For this research, we use an in-house wireless network simulator written in Java. The simulator constructs a field with variable number of nodes, communication radius, sink path and sink speed.

For each experiment, data was tabulated and averaged on one thousand randomly generated fields. These same thousand random fields were used for each graph.

We simulate a sensor field consisting of a rectangular field with dimensions of 200 by 200 units. Inside this X-Y range, a specified number of nodes are randomly placed in the field, each with a unique x-y coordinate. Each node has a communication radius that determines which nodes can communicate with each other. The default communication radius is 25 units unless otherwise specified. In this work we assume that all nodes are in radio range of at least one other node. Upon construction of the field, all nodes in the field must be connected, and the field will be reconstructed until it meets that requirement.

Once the field is successfully constructed, a receiver sink is constructed and either given a random linear path, or one of three specific nonlinear paths. All paths start from outside of the X-Y range and out of communication range of any node in the field. The sink speed is expressed in the distance covered per message sent.

After a target is selected, the sink will obtain a packet of information for every time ”x,” which is the send-receive time for node-to-node communication. The default value of x used for all simulations was 40 milli-units of time. The sink sends out a message every x calling for the next packet of data, which is recursively sent down the link tree. Packets are sent back as long as the sink is within range of the node. Once the sink is no longer within range, a new target is selected using the target node selection algorithm.

5. Results

We present the results of three design decisions: how the moving sink determines with which node to communicate, how to minimize the number of messages sent while maximizing the amount of data collected, and, how the sink path affects the system.

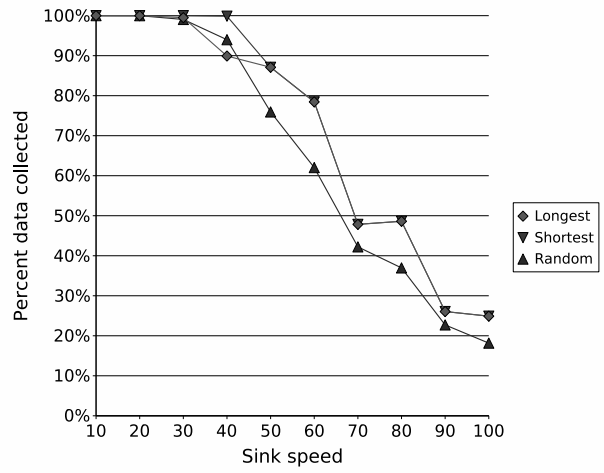

Figure 3. The percent data collected versus sink speed for each of the target node selection algorithms.

5.1. Target Node Selection

We study three selection mechanisms for choosing the target node from among those nodes in communication range.

A new target is selected every timer tick (1 unit of simulation time) that the previous target is no longer within communication range. Target selection is quite expensive because it changes the route messages will take, which leads to wasted messages going to the previous target node. *Shortest* and *longest* both keep track of how many timer ticks each of the current nodes in communication range have been in range; *shortest* selects the node that has been in range the fewest ticks, while *longest* selects the node that has been in range the most ticks. *Random* target selection randomly selects one of the nodes currently in communication range.

Figure 3 shows that at higher speeds, *longest* and *shortest* both perform well, followed by *random*. In addition, *longest* and *shortest* perform similarly at both very high and very low speeds. This is reasonable because at high speeds, each node in range is likely to only be within range for one timer tick, and both *shortest* and *longest* select the same node if all are in range for only one tick. Likewise, they are similar at slow speeds because the inefficiency of *longest* is overcome when it is in the field for a longer period of time. This means that at the slower speeds, *longest* approaches the upper bound of 100% data collection which *shortest* had approached earlier.

There is a middle range, however, at which *shortest* performs better than *longest*. This range changes depending on how large the communication radius is, but *shortest* always performs better than *longest*. At a radius of 25, the average percent data collected for *longest*, *shortest*, and *random* is 70%, 71%, and 65% respectively. As the radius increases, the speed at which *shortest* outperforms *longest* increases. This occurs because a larger communication radius benefits from good target selection. Good target selection is characterized by selecting a node which will in communication range for the longest period of time. A long duration for the target node is advantageous because it sends the data in the correct direction

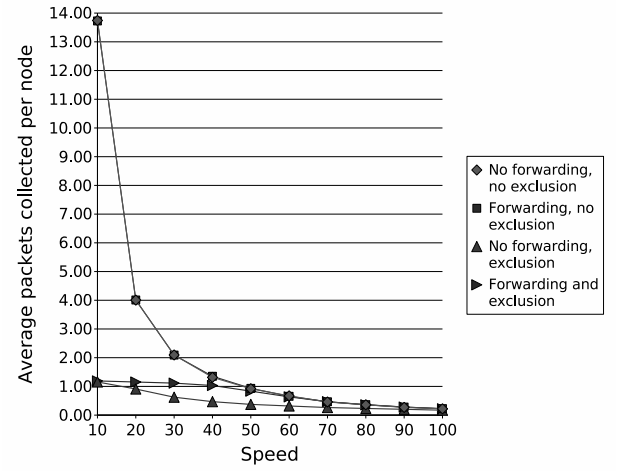

Figure 4. The average number of messages per node with the 4 possible configurations. Reducing redundant messages dramatically reduces the total messages in the system.

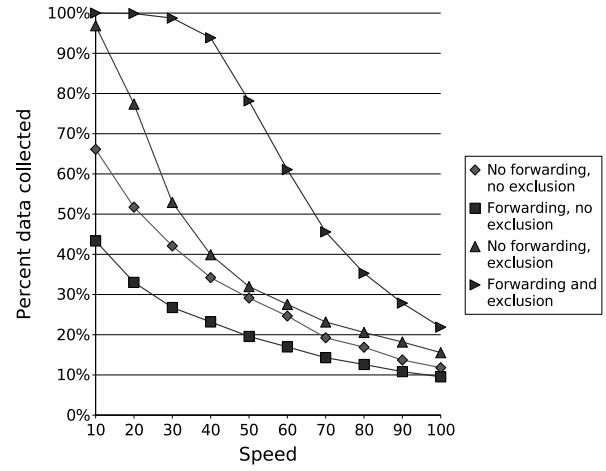

Figure 5. The percent of data received per node, varying data forwarding and message dropping.

for more time. *Shortest* tends to select nodes with longer durations because it selects the node that has not been in range for long. This makes it less likely that the sink is about to leave the communication range of that node. At a radius of 40, the average percent data collected for *longest*, *shortest*, and *random* is 77%, 84%, and 81% respectively.

Random target selection does not perform as well as *longest* or *shortest* at high sink speeds. Where *shortest* and *longest* separate in performance, *random* performs better than *longest*, but still not as well as *shortest*. Due to these results, we will assume the *shortest* algorithm for the rest of the experiments.

5.2. Optimizing Data Gathering

When a moving sink switches target nodes, there are several efficiency problems. The messages that leaf nodes have sent but did not reach the initial target node may need to be either eliminated or re-routed. Additionally, data that was not collected may need to be retransmitted. We present two complementary techniques which decrease the number of messages in the network and increase the amount of data collected.

5.2.1 Reducing Redundant Messages

To prevent redundant messages and increase the percent of the overall data that is collected, certain messages are excluded after having been received once. Originally, all messages were excluded after the first reception of a specific source and parent address combination, but this proved to require significant memory usage from each node.

To reduce the memory usage, only source and parent address pairs that have the same value for both source and parent are excluded. Figure 4 shows that even with this simple version of exclusion, average packets collected per node reduced from 2.41 to .73, with an increase of 19% in the percent data collected. This demonstrates how redundant messages were clogging the network, preventing unique messages from reaching the sink.

5.2.2 Data Forwarding

When a new target node is selected, messages in transit may not be routed to the new target node. If those messages are not re-routed, they need to be dropped to minimize the traffic through the network. Instead of dropping those messages, we can change the destination of the messages to the new target and continue routing those messages. The intuition is that the message is closer now than it was when it started, so this message forwarding will provide a net gain in data collected, even though there will be more messages total.

Figure 5 shows that with data forwarding, we see a dramatic increase in the percent data received. Figure 4 shows that there is only a small increase in total messages - in fact, those are the very messages we needed. The average packets collected per node now follows very closely the percent data collected, so we must be receiving very close to one message per node that gets its data sent, as opposed to many messages from some nodes and none from others.

One interesting configuration is when we have forwarding but no redundant message removal. The percent of data collected actually decreases because those forwarded messages add to an already clogged system without successfully delivering their data.

5.3. Sink Path Efficiency

In order to test the effectiveness of the algorithms, we run multiple nonlinear paths in addition to the linear ones. Figure 7 depicts the three nonlinear paths tested: “L” shape, “U” shape, and a third resembling a “sawtooth.”

In testing, U and sawtooth perform significantly better than L and linear. However, sawtooth and U are in the field for longer periods of time than L and linear. A sink that travels in such a pattern that it directly communicates with every node

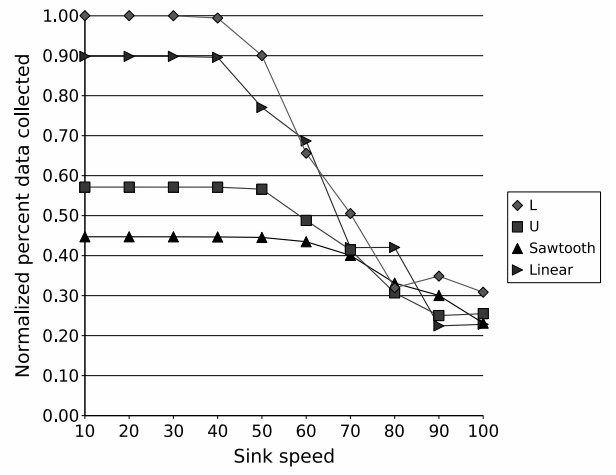

Figure 6. Normalized percent data collected per unit time spent in the field versus sink speed.

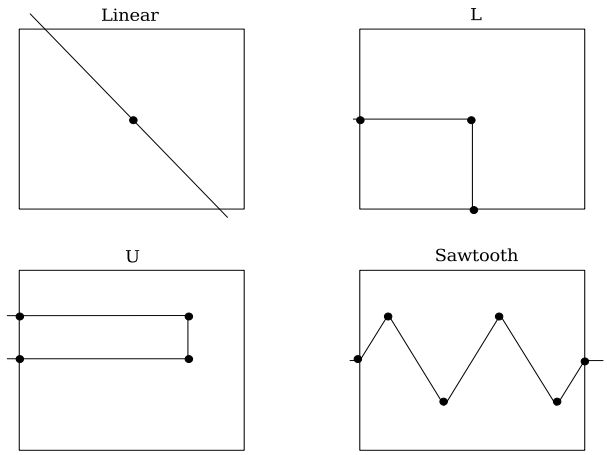

Figure 7. Various path simulations.

could gather 100% of the data, so absolute percentage is not the only relevant measure.

We also look at the normalized efficiency of the path, defined as the percent data collected per unit time spent in the field. Results for this experiment are shown in Figure 6. At lower speeds, the paths that spent less time in the field were more efficient. Because they continue in the same general direction, L and linear have fewer overall message re-routes than U and sawtooth.

In terms of efficiency, they all perform similarly at high speeds. Because they are all getting useful information each time they change targets, the effect of path is minimal. We find that, regardless of the path, all paths perform within 8% of the other paths. The only difference is that the U and sawtooth are actually gathering more total data because of increased time in the field.

6. Future Work

A functioning implementation of this algorithm is currently being developed using TinyOS as the operating system. MICA2 motes are being used for the sensor field nodes. By using two items from TinyOS, TOSSIM for testing and the WMEWMA multihop routine for a foundation, a system to determine the links and to forward messages has been created. A system to set the topography of the motes using a string of broadcast messages has also been developed.

7. Conclusion

Data retrieval from a wireless sensor network is a challenging task when the data collector (sink) is moving at high speed. In this work, we look at how the sink should choose a target node for communication, how the data should be routed to the sink in order to maximize the amount of data collected, and how the sink path affects data retrieval.

We find that routing data towards the node that has most recently entered the communication range of the sink produces good results. Selecting the least recent or a random target node did not perform as well, especially for larger communication radii.

For the process of actually sending the data, a combination of incremental message forwarding and redundant message exclusion collects 35% more data on average. We find these two techniques to be necessary and complementary. We find exclusion alone only collected 19% more data than the baseline and forwarding collected 10% less data.

Finally, we show that the effectiveness of this algorithm does not depend on the path taken through the field. While any path that is in the field for a longer period of time collects more data, results that were normalized for time spent on the field performed almost equally. Variations in the percent of data collected were within 3% on average, with a maximum of 8%.

References

[1] C. Intanagonwiwat. Impact of network density on data aggregation in wireless sensor networks. In *International Conference on Distributed Computing Systems (ICDCS-22), November*.

[2] Intuicom. Urban seismic monitoring network, 2004.

[3] P. Juang, H. Oki, Y. Wang, M. Martonosi, L.-S. Peh, and D. Rubenstein. Energy-effi cient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet. In *Architectural Support for Programming Languages and Operating Systems, ASPLOS-X*, 2002.

[4] B. Krishnamachari, D. Estrin, and S. Wicker. Impact of data aggregation in wireless sensor networks. In *Workshop on Distributed Event-Based Systems, DEBS'02*, 2002.

[5] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. Tag: a tiny aggregation service for ad-hoc sensor networks. In *Symposium on Operating Systems Design and Implementation, OSDI 02*, 2002.

[6] F. Ye, A. Chen, S. Lu, and L. Zhang. A scalable solution to minimum cost forwarding in large scale sensor networks. In *International Conference on Computer Communications and Networks, ICCCN 2001*, 2001.