Efficient Continuous Mapping in Sensor Networks Using Isolines

Ignacio Solis and Katia Obraczka
{isolis, katia}@cse.ucsc.edu
Computer Engineering Department
University of California, Santa Cruz

April 15, 2005

Abstract

This paper introduces an energy-efficient data collection technique that takes advantage of spatial/temporal data correlation to generate maps for continuous monitoring (e.g., of environmental conditions such as temperature, humidity, etc.). In its essence, the proposed technique, *isoline aggregation*, works by detecting *isolines* which are the lines that make up a contour map. Energy efficiency through spatial aggregation is achieved by having only nodes that detect the isoline report to the sink. Simulation results show that *isoline aggregation* can reduce the amount of bytes transmitted by a factor of 11 compared to when no data aggregation is used and by up to 4 times when compared to an existing spatial-correlation based aggregation mechanism. At the same time, we are able to keep high data accuracy.

1 Introduction

In-network aggregation has been employed quite successfully as an effective energy savings technique in power-constrained, data-driven sensor networks. The main idea behind in-network aggregation is to process data as it flows from sensor nodes to information sinks.

A number of existing aggregation algorithms focus on monitoring applications (e.g., environmental monitoring). *Directed diffusion* [1], some of the aggregation mechanisms proposed by Tiny Aggregation (TAG) [2], and *cascading timers* [3] are some notable examples. They process data, which is generated by sensors periodically, as it flows over a data collection structure (e.g., tree) rooted at the sink and spanning all relevant sensors. Our work on *cascading timers* [3] shows that the timing model used by the aggregation algorithms is critical to achieve energy efficiency without sacrificing data freshness.

In this paper, we describe a novel aggregation technique that targets spatially and temporally-correlated data. In particular, we address applications that are continuously monitoring varying conditions of a given geographic region (e.g., temperature, rain fall, radiation, etc.) and, as a result, generate a "contour map" of the sensed variable.

The proposed algorithm, or *isoline aggregation*, takes advantage of the spatial correlation of data in these monitoring scenarios and group nodes that report similar readings into *isoclusters*. Analogously to contour maps, an isocluster is represented by an area of the same value (color) delimited by an *isopleth*. An *isopleth*, (from the Greek *iso* - same, *pleth* - value), or *isoline*, is defined as a line connecting points of the same value.

Energy efficiency is achieved by, instead of having all nodes send their readings to the sink, have only a few nodes per isocluster report to the sink. For instance, if we were to generate a contour temperature map, an isotherm, once the temperature ranges are defined, nodes would be grouped into areas that exhibit temperatures within the defined ranges. To construct an isotherm we do not need to collect data from all the nodes in the region being monitored; it is sufficient to collect the isolines, draw them and "color in" the corresponding areas. This is how energy efficiency is attained.

Through simulations, we evaluate *isoline aggregation* and compare it against no aggregation and our implementation of aggregation using polygons. *Polygon* aggregation is the general approach used in previous approaches by e-Scan [4] and *isobars* [5], both of which are discussed in more detail in Section 2 below. Our results show that *isoline aggregation* can achieve significant energy savings when compared to no aggregation and *polygon* aggregation, while yielding high data accuracy.

The remainder of the paper is organized as follows. *Isoline aggregation* is described in Section 3. Our experimental methodology is presented in Section 4 followed by a discussion of simulation results in Section 5. Section 6 presents our concluding remarks and directions of future work.

2 Related Work

There are mainly two other approaches that target spatially-correlated data aggregation for mapping, namely *eScan* [4] and *isobars* [5]. *eScan* focuses on monitoring the sensor network itself, in particular the remaining energy in the nodes. It queries sensing nodes which, in turn, report their remaining energy. This is done via a data collection tree established at query propagation time. When the data is being reported back to the sink, nodes aggregate the information as it flows. Aggregation is done by grouping readings that meet a certain criteria. In particular, for a set of readings to be aggregated, they need to be geographically adjacent and they need to be in the same value range.

Data is aggregated into polygons of similar value and represented by the corresponding polygon's coordinates. This approach has a few drawbacks. For one, the aggregation is done as the data flows down the collection tree, which is not always the most efficient way. For instance, if two nodes close-by are in the same value range but are in different branches of the tree, their values are not aggregated until they reach a common ancestor, who may be further up the tree. Also, for a node to aggregate the coordinates of a polygon it needs to know the exact location of the nodes. Assuming geographic location encoding requires more bytes than node identifiers, propagating location information results in significant additional overhead.

In contrast, *isoline aggregation* employs localized aggregation by detecting *isolines* and neighboring nodes. Hence, aggregating down the path is not necessary. Location information is only needed at the sink and can be collected once.

The *isobar* mapping approach is part of the the advanced aggregation techniques proposed in TAG [5]. Here nodes are part of a grid. A node's location is based on its position on the grid. Data is collected by aggregating nodes with similar readings into polygons. On a heavily populated grid, aggregation yields good results. If the grid is sparse, or if packets are dropped, or if energy efficient is favored over accuracy, *bounding boxes* are used for defining the polygons. A bounding box is created around an area to be aggregated. Cuts are then made to the bounding box to approximate the shape of the polygon. The more cuts, the more data that needs to be reported and the better the accuracy. Less cuts means decreased accuracy, but less data to be sent, improving energy efficiency.

Isobar mapping suffers from similar problems as *eScan* since it also performs aggregation as the data flows towards the sink. Node location is represented by the corresponding grid coordinates minimizing the need to transmit real location information.

More recently, the energy-accuracy tradeoff study by Boulis et al. [6] proposes a data collection mechanism where nodes decide whether to share their own readings based on estimates they get from other nodes. While this works well for operations like reporting the maximum or minimum value, it does not apply to more complex applications like mapping.

Approaching the subject from a mathematical angle, Doherty et al. [7] studied how different mechanisms to collect scattered data perform in dense sensor networks. The focus of their work is on node selection rather than a protocol to achieve it.

Both *eScan* and *isobars* use polygons to aggregate data of similar value produced by neighboring nodes. We will compare the performance of *isoline aggregation* against (1) no aggregation and (2) *polygon* aggregation, our implementation of the aggregation mechanism underlying both *eScan* and *isobars*. Even though continuous monitoring is not specifically addressed by *eScan* or *isobars*, in our implementation of *polygon* aggregation we employ temporal aggregation so we can conduct a fair performance comparison against *isolines*.

3 Isolines

The goal of *isoline aggregation* is to provide energy-efficient data collection by reducing redundant transmissions. One challenge is to achieve this goal using only local information. Another challenge is to maintain high data accuracy. To address these challenges, *isoline aggregation* uses the concept of *isolines*, lines of the same value. Energy efficiency is achieved by having each node only report to the sink if it detects an isoline between itself and its neighbors; otherwise, no report is generated.

3.1 Neighbor-to-Neighbor Protocol

Isolines are detected based on neighborhood information gathered through a neighbor-to-neighbor protocol, or NNP. Essentially, NNP broadcasts the local sensed value. Nodes decide when they need to communicate their sensed information to their neighbors. This happens when (1) a node is started and (2) when data changes cause an isoline to appear or disappear. First-time reports allow the network to detect initial values and the presence of new nodes.

3.2 Isoline Detection and Reporting

Isoline detection is a very simple yet elegant method of collecting information efficiently for contour map generation. It works as follows. First, a node compares its reading with the reading of all neighboring nodes. If the readings lie in different sides of an isoline, then a report needs to be generated. For example, if the isolines measure multiples of 10, then a node sensing 35 and a neighbor whose sensed value is 42 are able to detect that there is (at least) one isoline of value 40 passing between them.

Once the existence of an isoline has been determined, it needs to be reported to the data collection sink. Reporting an isoline consists of sending to the sink the node's sensed value and the value of the neighboring node across the isoline. Sending just the isoline value saves some bytes at the expense of accuracy.

The detection of the isoline is symmetric, i.e., both the node and its neighbor will detect it. The node who reports the isoline is the one closest to the sink (according to hop count). If both nodes are at the same distance, the node with the lowest reading will send the message.

Nodes will only report to the sink when there are new isolines nearby. This could lead to problems since we assume that, on the absence of reports, nothing has changed. For this situation we also implement probabilistic reporting. Nodes broadcast their information periodically even when they do not need to do so. This also helps improving the accuracy of the maps generated and can be fine-tuned appropriately.

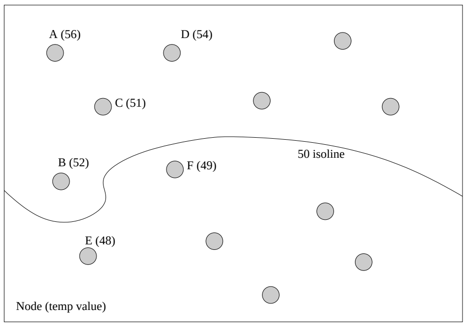

Figure 1: A temperature isoline.

Nodes that are inside the same isoline are said to belong to the same *isocluster*. Figure 1 depicts a temperature isoline where temperatures within 10-degree ranges belong to the same isocluster. Nodes *A* and *C* are part of the same isocluster and will know, upon exchanging sensed values, that an isoline does not exist between them. On the other hand, nodes *B* whose value is 52 and *E* who is measuring a temperature of 48 will detect the 50-degree isoline when they compare values.

There is a clear trade-off between the range of the isolines which determines the accuracy of the aggregation algorithm and its energy efficiency. The smaller range an isoline covers, the more accurate the overall map is. However, denser contour maps (in terms of number of isolines per area), generate more data and thus are less energy efficient.

3.3 Continuous Monitoring

```
while(1){
  get_reading_from_sensor()
  if(reading_has_changed_range){
    broadcast_reading_to_neighbors()
  }
  while_monitor_events {
    switch(event){
      case receive_reading_from_neighbor:
        if(reading_ranges_differ()){
          //determine who sends the readings
          //to the sink, me or the neighbor
          if(i_am_closer_to_sink ||
             (we_are_at_same_distance &&
              my_reading_is_lower)){
            add_readings_to_report()
          }
          //else
          //  neighbor will send readings
        }
        //else
        //  we are in the same range so we
        //  don't report
        break;
      case period_over:
        exit_event_monitoring()
    }
  }
  if(report_contains_readings){
    send_report_to_sink()
    reset_report()
  }
}
```

Figure 2: Continuous isoline main loop pseudo-code

In applications that are continuously monitoring a specific condition, the sensor network needs to continuously sample sensing nodes and report the current state to the sink. To accomplish this task in an energy-efficient manner, *isoline* aggregation takes advantage of temporal data correlation since it reports only when changes are significant (ie. isolines move).

This capability is based on the normal isoline detection mechanism. More specifically, a node starts by sampling its sensor and using the NNP to communicate its reading to its neighbors. Based on the information it gets from its neighbors, a node will report to the sink if an isoline is detected. Nodes will then sample the sensor periodically, but will broadcast their reading only when the value changes from one range to another. For example, if sensors are monitoring ranges of 10s (0-9,10-19,20-29,...) and a node's reading goes from 17 to 23, the node sends out a NNP broadcast. This basically means that an isoline moved, appeared or disappeared, and the node needs to check if it needs to report.

The pseudo-code for the continuous isoline main loop algorithm can be seen in Figure 2. The neighbor to neighbor protocol is simply the broadcast_reading_to_neighbors(). This message contains a node's reading, the node id and its distance (hop count) to the sink. For clarity the pseudo-code leaves out some error detection features.

Isoline aggregation uses cascading timers [3] to schedule node transmissions as a function of the node's position in the data collection tree. The outcome is that a cascading effect is achieved from the leaf nodes to the data sink; in other words, the farthest away node schedules its transmission first; the next hop (toward the sink), schedules its transmission next allowing enough time to receive data from its children, etc. Not only does this reduce delay, but it fits very well with schemes that save energy by turning off the radio when the node is idle.

4 Experimental Methodology

We use simulations to evaluate the performance of *isoline aggregation* and compare it against no aggregation and polygon aggregation. The remainder of this section presents our experimental methodology in detail, including descriptions of the scenarios and simulation setup used.

4.1 Other Aggregation Algorithms

Polygon aggregation is used to represent the existing algorithms of *eScan* and *isobars*. Our implementation of is as follows: when a node receives data from its children, it aggregates it into polygons and sends only the polygons' vertices. All nodes in the polygon are assigned the average value of the polygon's range; for example, in the case of a [40-50) temperature range, nodes take the value of 45. Similarly to *isobar* aggregation [5], we use the sensors' grid coordinates as their locations. Reports from nodes may contain multiple polygons if multiple value ranges have been aggregated. *Polygon aggregation* uses the PolyBoolean library [8] for handling polygons and aggregating them. For both *polygon*- and *isoline aggregation*, data is grouped in ranges of 10, that is, from [0-10),[10-20), etc.

Just like isolines, we use *cascading timers* [3] as the timing model since it allows nodes to wait for their children to report without increasing the data collection delay. This is important because we are doing continuous monitoring and need to deliver information in a timely fashion.

The original *eScan* and *isobars* algorithms did not provide continuous monitoring. For fairness reasons, we incorporate temporal aggregation when extending *polygon aggregation* to perform continuous monitoring. For instance, if the temperature of a node has not changed ranges from the last time this node reported, a report will not be generated. This implies that leaf nodes will not report if there has been no range change since the last report. Inner tree nodes will have to report if leaf nodes report since they are part of the aggregation tree. Polygon aggregation needs them to include their information to perform aggregation.

No aggregation is our baseline algorithm. Nodes simply send their readings down the aggregation tree to the sink. This is very simple and effective. Basically the sink will get a reading from every node and will be able to generate the most accurate map using all possible sensed values. However, there are various problems with this approach. Having all nodes report means that considerable traffic will be flowing through the network.

No Aggregation Optimized builds on *No aggregation* by using temporal data correlation to reduce the number of reports generated. In this optimized version, nodes report their readings directly to the sink only when their temperature has changed from one range to another.

4.2 Scenarios

In our experiments, we try to model two types of phenomena: hotspots and moving fronts. For our first scenario, we generate a hotspot in our map with a temperature increase of 25 degrees. Temperature increase decays at a power of 4 from the center of the hotspot. The region will then slowly go back to the original temperature, as the hotspot vanishes. In our second scenario, we have a front moving in from the left to right. Temperature increases from the forties to the seventies in about 150 meters. The front moves to the right in 13 seconds.

4.3 Simulation Setup

As the experimental platform, we employ the `ns-2` [9] network simulator. We use a sensor network consisting of $16x16$ nodes arranged in a evenly spaced grid monitoring temperature in a $400m^2area$. We should point out that, although, in these experiments, nodes are placed according to a grid pattern, *isoline aggregation* is not specific to

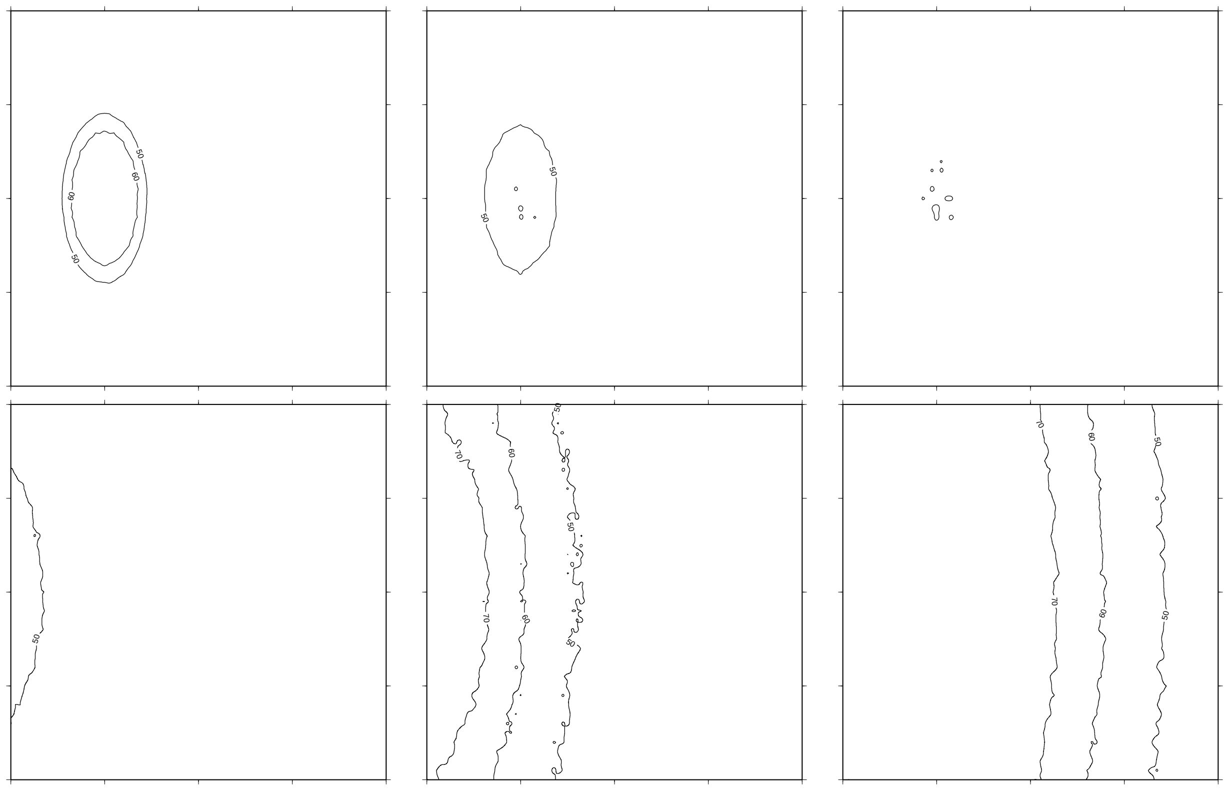

Figure 3: Hotspot and front scenarios.

grid placement. For random node placement, further optimization for more careful determination of which nodes should be reporting is needed to avoid unnecessary redundancy.

For medium access control, nodes use CSMA at 115Kbps. Their transmission range is set to 40m. FLIP [10] was used as the network protocol. Node identifiers and location information are 2 bytes long. Temperature information is also 2 bytes. Nodes sample the environment in rounds every 1 second.

The experiment collects the temperature information of the area for a period of 15 seconds. Figure 3 shows the real maps for our two scenarios at different points in time. Reality is simulated by a matrix of $80x40$ points. This will represent the area we are monitoring. When a node located at the center of our area samples its sensor it will read the center value of the matrix. These values are determined by the scenario and time in the simulation. The starting value of all points is centered at 45 degrees with a small randomness factor of +/- 2. The sensor network will be basically subsampling this matrix of 80X40 with a maximum of 16X16 samples if all nodes are reporting.

The sink node, which is placed at the center of the map, starts by broadcasting a query for the map at time 1s. From time 3s to 14s, nodes report their temperature readings. The simulation is stopped at time 15s. The data points used to compute the tabulated results in Section 5 are obtained by averaging over 10 runs.

4.4 Performance Metrics

With these considerations in mind, the performance of the aggregation algorithms is evaluated according to two metrics: their energy efficiency and their accuracy. Of course, our goal is to minimize energy consumption without sacrificing data accuracy. We use average number of bytes transmitted by each aggregation protocol to measure energy consumption. This includes all data transmitted by all nodes, intermediate nodes included. While this might seem to not take receive power into consideration we should note that the timing scheme used, cascading timeouts, is well suited to MAC protocols that switch idle nodes off to low-power radio mode since communication is not taking place. This is true for both polygon aggregation and isoline aggregation. No aggregation doesn't do scheduling of transmissions and cannot use such a scheme without additional modifications.

Accuracy is measured by how **similar** the map produced is to reality. Since our goal is to draw contour maps, it seems appropriate to calculate similarity as the percentage of points that are actually in the correct value range when compared to reality. That is, for every point, we compare the reality value to the value interpolated from the data collected in a specific round. We calculate the percentage of points in the correct value range and then average it over all simulation rounds (seconds). Simulations were run 10 times. This metric is called *contour similarity*.

5 Results

Plotting results obtained from the simulations helps us visualize how the aggregation algorithms perform in terms of accuracy. We should point out that the graphing tools we use, which interpolate the data points to generate a map, have not been optimized for aggregation algorithms. For example, in the case of *isolines*, if there are no readings received from an area, then it is an indication that an isoline does not exist. In the case of *polygon aggregation*, reported areas should be graphed as polygons. Moreover, some of the anomalies in the graphs can be caused by lost packets. However, since we do not insert artificial drops, the effects are not very evident. Packet losses can still occur caused by collisions.

To generate the maps we interpolate the missing points from the data the sink received. We use the `ngmath` part of the NCAR Graphics Library[11] to interpolate the $80x40$ reality data points. As previously pointed out, the assumptions made by the interpolation algorithm may not be adequate to the data collection scenarios under consideration. For example, if we are capturing a gradient, like the one in the moving front scenario, some of the collection algorithms will only send the points corresponding to the nodes that changed value. If we only provide these points to the interpolation algorithm, it will assume that the gradient continues on both sides of the front. This will create very disparate graphs. The solution to this problem is to use the last data reported by a node as input to the interpolation. This problem was very pronounced on the "optimized" version of no aggregation.

In the case of *isoline aggregation* we have, by definition, nodes reporting only when they detect an isoline. Hence, we can infer that if no node reported, the values of that area have not changed over to the "next" isoline. So, if the highest report is a 46, we know that if the interpolation algorithm generates a value greater than 50, that value should not be considered.

	Similarity	KBytes sent
No Agg.	98.7 (sd 0.09)	180.0 (sd 5.4)
No Agg opt.	98.9 (sd 0.09)	21.1 (sd 0.4)
Polygons	98.1 (sd 0.49)	62.9 (sd 4.6)
Isolines	97.0 (sd 0.36)	15.3 (sd 1.2)

Table 1: Hotspot scenario: contour similarity and number of bytes sent.

Tables 1 and 2 present results for the hotspot and moving front scenarios, respectively. The first column shows contour similarity and corresponding standard deviation (in parenthesis). The second column shows the average number of kilobytes sent per round. This measures potential energy savings.

For the hotspot scenario we observe that all algorithms exhibit very good accuracy. This is to be expected as there are not many nodes changing values. The greatest differences can be seen in the amount of data sent. The unoptimized version of no aggregation sent a total of 180KB. That is more than 11 times the amount of data sent by *isoline aggregation*.

No aggregation optimized sends a total of 21KB. That is 38% more than *isoline aggregation*. Yet, isolines yields similar accuracy. This difference is mostly due to the fact that the initial collection without using aggregation is very expensive. When data changes in groups, *isoline aggregation* exhibits lower overhead since nodes with neighbors in the same range will not need to report. Therefore, if larger areas change values (e.g., if the hot spot affects larger regions), we expect that the energy savings achieved by *isoline aggregation* will be even more pronounced.

In the case of *polygon aggregation* more data is sent because aggregation happens only down the collection tree, and this implies that nodes which would otherwise not have reported will report. It also means that aggregation does not take place as soon as possible.

Figure 4 shows the maps generated from the data collected at time $T = 4$ for the hotspot scenario. This can be compared to the real map in Figure 3. Note that the maps in Figure 4 are a snapshot at time $T = 4$ of a single run; therefore, nuances and quirks are to be expected. We can see that the maps from *Isoline* and *None* (i.e., no aggregation without optimization) are the ones that most resemble the reality map.

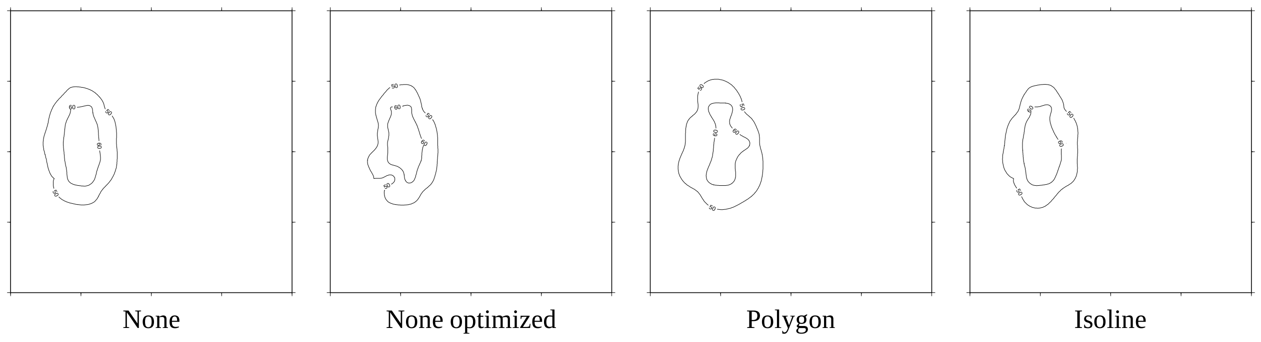

Figure 4: Hotspot scenario, T=4.

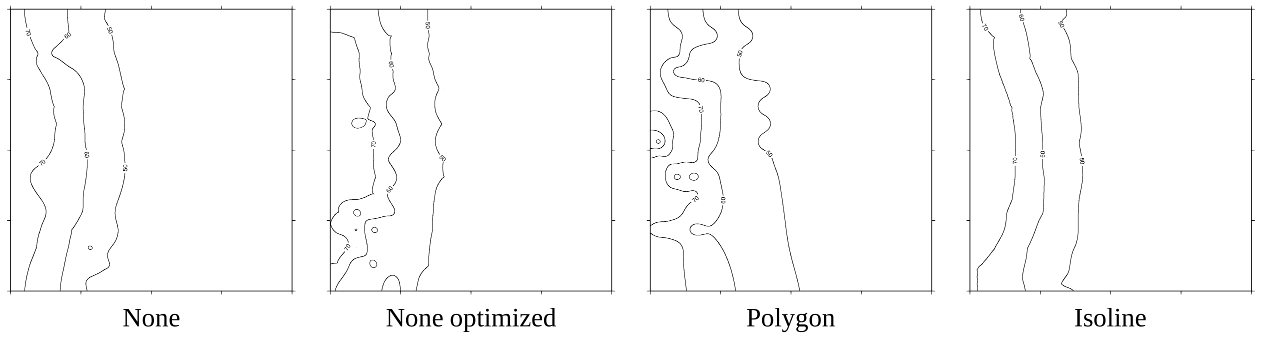

Figure 5: Front scenario, T=7.

	Similarity	KBytes sent
No Agg.	93.2 (sd 1.72)	177.1 (sd 5.9)
No Agg opt.	89.3 (sd 0.70)	62.1 (sd 2.3)
Polygons	82.4 (sd 2.93)	77.0 (sd 3.6)
Isolines	96.7 (sd 0.50)	55.8 (sd 3.1)

Table 2: Front scenario: contour similarity and number of bytes sent.

Results for the moving front scenario are presented in Table 2. In this scenario, all nodes will eventually change value. The amount of data that needs to be reported is thus larger.

In terms of accuracy, *isoline aggregation* comes on top, with 96.7% similarity. Sending less data causes less contention at the MAC layer and therefore is able to achieve better accuracy. Note that this is true even if an underlying reliable MAC protocol were used. In fact, depending on the the reliability mechanism used by the MAC layer, higher contention and higher delays can be incurred.

The other optimized methods, i.e., "optimized" no aggregation and *polygons*, both have accuracy in the 80%s while still transmitting more data than *isolines*. The unoptimized no aggregation method exhibits 3.5% less similarity than *isolines* yet reporting over 3 times more data.

Figure 5 shows maps generated by the algorithms for the moving front at time $T = 7$. In the case of *isolines* and "unoptimized" no aggregation, the resulting graphs are relatively accurate. There are still some minor mistakes which happen from reports that got lost along the way. In the case of "optimized" no aggregation, the errors are different: they are "carry overs". Since nodes only report when data changes, and the map is constructed using old readings, a node whose update is lost might stay with an old value and create a semi-permanent error in the graph.

In the case of *polygon aggregation*, when a packet is lost, we loose an aggregated report, and hence we see areas where there are problems. *Isoline aggregation* reports quite accurately with some minor distortion near the edges due to the lack of nodes to sample reality and the interpolation algorithm.

It is important to note that the algorithms presented here might not be able to get 100% similarity under the contour similarity metric. This is because a perfect score implies that all of the reality points are placed in the correct ranges. This might not be possible since we are only subsampling the space. At some point it is necessary to increase sensor density to obtain higher accuracy. We expect that higher sensor density will help *isoline aggregation*'s accuracy. As pointed out previously, accuracy increases at the expense of energy savings. However, *isoline aggregation* minimizes this trade-off.

One of the disadvantages of using *polygon*-based aggregation lies in the fact that a node cannot make a localized decision of whether or not its information is redundant. Aggregation is limited to nodes on the same branch of a tree and only happens when a shared ancestor exists. If this ancestor node is far away, redundant information will propagated closer to the sink.

6 Conclusions and Future Work

Continuous monitoring is one key application of sensor networks. This paper introduced *isoline aggregation*, an energy-efficient data collection technique targeting continuous mapping in sensor networks. By having nodes monitor the presence of *isolines* and report when they appear, *isoline aggregation* is a simple, elegant, and efficient algorithm for generating accurate maps of continuously changing conditions.

Through simulations, we evaluate the performance of *isoline aggregation* in terms of its accuracy and energy efficiency. Two different temperature monitoring scenarios were simulated, namely hotspot and moving front. In the hotspot scenario, a spot of activity appears and disappears; in the front scenario, temperature increase wave moves through the area. Both of these scenarios have real life analogies (e.g., fires, animal migration).

In the hotspot scenario, *isolines* delivered comparable accuracy while sending 38% less data than our nearest competitor, an optimized form of no aggregation, where nodes report only when the sensed value changes. The non-optimized alternative required more than 11 times the amount of data transmitted. For the front scenario, *isoline aggregation* delivered higher accuracy than all other protocols we studied (including *polygon aggregation*, which represented schemes like *eScan* and *isobars*, while still yielding the highest energy efficiency.

We plan to continue our work on mapping techniques using *isolines*. We will use node placement policies other than grid placement will also study the effect of positive acknowledgement aggregation. The effect of range size on accuracy is also being investigated. We will also develop and evaluate *isoline aggregation* on a small sensor network testbed consisting of Berkeley Motes.

References

[1] C. Intanagonwiwat, R. Govindan and D. Estrin: Directed diffusion: A scalable and robust communication paradigm for sensor networks. In: Proceedings of the International Conference on Mobile Computing and Networking (MobiCom), ACM (2000)

[2] S.R. Madden, M.J. Franklin, J.M. Hellerstein, and W. Hong: TAG: a tiny aggregation service for adhoc sensor networks. In: Proceedings of the Symposium on Operating Systems Design and Implementation, OSDI. (2002)

[3] Solis, I., Obraczka, K.: The impact of timing in data aggregation for sensor networks. Proceedings of the IEEE International Conference on Communications (ICC) (2004)

[4] J. Zhao, R. Govindan and D. Estrin: Residual energy scans for monitoring wireless sensor networks. In: Proceedings of the IEEE Wilress Communications and Networking Conference (WCNC'02). (2002) 17–21

[5] J. M. Hellerstein, W. Hong, S. Madden and K. Stanek: Beyond average: Towards sophisticated sensing with queries. In: Proceedings of the 2nd International Workshop on Information Processing in Sensor Networks (IPSN '03). (2003)

[6] A. Boulis, S. Ganeriwal, and M.B. Srivastava: Aggregation in sensor networks: An energy-accuracy trade-off. In: Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications. (2003)

[7] L. Doherty and K. S. J. Pister : Scattered data selection for dense sensor networks. In: Proceedings of the Third Symposium on Information Processing in Sensor Networks. (2004)

[8] Leonov, M.: PolyBoolean Library (2004) http://www.complex-a5.ru/polyboolean/.

[9] VINT: The Network Simulator NS-2. http://www.isi.edu/nsnam (2001)

[10] Solis, I., Obraczka, K.: FLIP: A flexible interconnection protocol for heterogeneous internetworking. ACM/Kluwer Mobile Networking and Applications (MONET) Special on Integration of Heterogeneous Wireless Technologies (2004)

[11] National Center for Atmospheric Research: NCAR Graphics Library (2004) http://ngwww.ucar.edu/ng4.3/.