Title
Contour maps: Monitoring and diagnosis in sensor networks

Journal
Computer Networks, 50(15)

ISSN
1389-1286

Authors
Meng, Xiaoqiao Q
Nandagopal, T
Li, L
et al.

Publication Date
2006-10-01

Peer reviewed

Contour Maps: Monitoring and Diagnosis in Sensor Networks

Xiaoqiao Meng*, Thyaga Nandagopal‡, Li Li‡, Songwu Lu*

*Computer Science Dep., University of California
Los Angeles, CA 90095, USA
{xqmeng,slu}@cs.ucla.edu

‡Bell Laboratories
Holmdel, NJ 07733, USA
{thyaga,erranlli}@lucent.com

Abstract

Large-scale sensor networks impose energy and communication constraints, thus it is difficult to collect data from each individual sensor node and process it at the sink. In this paper, we propose an efficient data-collection scheme that can be used for event monitoring or network-wide diagnosis. Our scheme relies on the well-known representation of data – contour maps, which trade off accuracy with the amount of samples. The scheme consists of three novel algorithms to build contour maps: distributed spatial and temporal data suppression, contour reconstruction at the sink via interpolation and smoothing, and an efficient mechanism to convey routing information over multiple hops. By reducing the number of transmissions required to convey relevant information to the sink, the proposed contour mapping scheme saves energy and improves network lifetime. In a sharp contrast to related work in this area, the scheme *does not require all nodes to explicitly share information*.

The contour mapping scheme can be applied for tasks such as: (1) presenting a global picture of the network in both temporal and spatial domains. (2) being used as a diagnosis tool, e.g., to detect faulty sensors and to scan for residual energy. (3) working in concert with in-network aggregation schemes to further reduce the communication overhead of aggregation schemes. The proposed scheme imposes little processing and storage overhead, allowing for the sensor networking paradigm of ”dumb sensor, smart sink” which enables economical deployment of large-scale sensor networks.

Simulation results show that our scheme is resilient to both high packet loss rate and measurement noise. The design is also energy efficient, resulting in up to an-order-of-magnitude power savings when compared with the base line scheme where every sensor sends its report to the sink.

I. Introduction

Wireless sensor networks enable users to interact with the physical environment at an unprecedented level. When an event occurs , e.g., a sudden temperature change, the nearby sensors report their observations of the event back to a data collection center (called sink). Emerging applications include temperature and humidity monitoring, forest fi re alarms in national parks, seismic monitoring, structural response, etc.

In this paper, we address the problem of monitoring and diagnosis for sensor networks with emphasis on applications that have a good degree of spatial correlation for monitored events. We seek to effectively monitor events covered by the sensor networks. We also examine the network health (e.g., the current energy level of sensors) and fault diagnosis (e.g., when part of the network are compromised by malicious attacks) issue, which is critical for a proper operation of sensor networks.

There are four main challenges for an effi cient monitoring and diagnosis design. First, an event may trigger widely varying readings at sensors spread over a large area. In many cases, it is desirable to have a global view of the entire sensor fi eld regarding that single event, and to monitor the change on a temporal basis. For example, when fi re bursts out in a building, it might be necessary to continuously monitor all the sensors to observe the spread of the fi re and track its intensity. The second challenge is that a single digest (e.g., min, max, avg) may not be enough for describing an event. For example, multiple events can cause interference to create various highs and lows at points different from the actual event locations. Third, certain events are time-critical and we do not need readings from every sensor to respond to the event. It is more important for the sink to collect coarse-grained information on a timely fashion than fi ne-grained information with signifi cant latency. In certain scenarios, it is critical to know approximately how far the fi re spreads and react promptly. Lastly, sensor readings are affected by noise, and some sensors could even provide faulty readings. This requires the solution to be be robust against faulty or noisy readings.

In order to develop a data-collection scheme that meets the above challenges while still be communication and energy efficient, we introduce the notion of *event contours*. In general, a contour is a curve in a map that connects points with equal values. Neighboring contours have different values, and these values are separated by a pre-determined threshold. Contours can represent various events, such as altitude, temperature, concentrations, velocity, etc. In addition, a single map can represent various contours, where we can have altitude contours overlapping with concentration contours. In this paper, we demonstrate that having an event contour of a sensor field can be extremely useful for monitoring and diagnosis tasks in sensor networks, while conveying event information in a reasonably accurate manner.

Our contributions are two fold: (1) we propose two novel algorithms: a distributed spatial and temporal data suppression algorithm performed by individual sensor nodes, and a contour reconstruction algorithm performed by the sink. (2) We give a novel and efficient algorithm for conveying routing information that enables multihop local suppression. The algorithm is based on the use of Bloom filter and the knowledge of geographic locations of sensor nodes.

The rest of the paper is organized as follows. We briefly describe network model and assumptions in Section II. We give an overview of our scheme in Section III. We then describe the related work In Section IV. We describe details of the scheme in Section V and discuss important extensions in Section VI. We present several application scenarios in Section VII and evaluate the proposed scheme in Section VIII. We conclude the paper in Section IX.

II. Network Model

We consider a network of *fixed sensor nodes* that are deployed in a 2 or 3 dimensional space. A set of monitoring nodes, defined as *sinks*, are responsible for collecting data reports from sensor nodes. Each sensor node has four components: a sensory transducer(s), a radio transceiver, a power unit and a processing unit. Certain nodes in the network may possess only the latter three components: these are relay nodes meant to process and pass information from other sensors to the monitors. We assume that a heterogeneity of transducers can exist in the sensor network, and that most sensors have limited computational power and storage space. We do not make any assumptions on the node density of the network, except that events are sensed by more than one sensor.

We assume that knowledge of sensor node locations is available at the sink. The location information need not be precise. It could be computed even after deployment, using techniques such as [10]. We do not assume any specific routing or medium access protocol in this network. Our contour mapping algorithm requires only the existence of a routing algorithm which guarantees that packets generated by the sensors have at least one route to reach the sink.

III. Problem description and design overview

Our goal is to enable efficient event monitoring and diagnosis in sensor networks. For many such tasks, it is sufficient if only a subset of sensors respond. For example, in order to monitor residual energy of sensors, it is not necessary for all sensors to report their remaining energy level. On the other hand, it might not be sufficient to report a few aggregated values such as average, min, max or sum. The sink needs to have an approximate view of the spatial distribution of the energy dissipation over time, i.e. the spatial and temporal energy consumption trend in the network.

Revisiting the fire monitoring application mentioned in Section I, computing aggregates over regions could potentially give a rough idea of the distribution over multiple regions. However, defining regions appropriately is very critical and it depends on the application and the geography of the sensor field and has been acknowledged to be a difficult problem [6], due to the computation costs. It is not clear if defining regions and computing aggregates can still provide the desired granularity and flexibility in monitoring events in the sensor field.

In order to reduce resource consumption as much as possible and provide an overview of the entire network with a desired granularity, we seek an efficient data representation that allows us to trade off accuracy and communication cost. More importantly, we would like to achieve this with minimal *computation* and *collaboration* requirements from individual sensors. This allows us to avoid the overhead of coordination (e.g. synchronization, clustering) and achieve tasks with "dumb" sensors [12]. Dumb sensors can be mass-produced and thus enable economical deployment of large-scale sensor networks.

We first elaborate on the concept of *contour maps* that we use for data representation, we then give an overview of the proposed contour construction algorithm.

A. Data Representation using Contour Maps

Using contour maps to represent the distribution of data is a well-known technique when there exists a trade-off between having more information about the data and affording higher cost to collect the additional information. Contours, in essence, are lines that connect data points of equal value. Non-intersecting lines have different values. Contour maps have a *step-value* which separates two adjacent lines. For example, given a step-value of 10, two adjacent lines on a topographical contour map indicate points that differ by 10 units in height, while an isotherm contour map indicate regions that differ by 10 units in temperature. More contour lines indicate fine-grained data, which comes at the cost of collecting more information. Contour maps present a simple way of fine-tuning the trade-off between information and the cost of obtaining it by adjusting the step-values to suit situational requirements. There can be many contour maps, each corresponding to a different parameter for e.g., temperature, pressure, wind speed, with different step-values for each parameter, thereby giving a clearer picture of events in the sensor network. The step value for each contour can also vary depending on the application requirement, e.g. an application may want more resolutions for temperatures above a certain value.

B. Contour Map Construction and Its Applications

The concept of contour maps is simple and well-known. However, it is very challenging to construct contour maps in a sensor network setting. How do we decide which sensor's value does not need to be transmitted in-network? Can we do this without explicit coordination among sensors? If there is no explicit coordination, can the sink construct the contour with reasonable accuracy? We seek to address these challenges.

We utilize the notion of step-values in contours, to define a desired margin of error. A sensor need not transmit if its sensor reading can be deduced with bounded error. This can be achieved through spatial and temporal suppression techniques. The idea behind spatial suppression is to exploit the correlation between neighboring sensors when an event occurs. By overhearing the shared medium , each sensor u determines if it should transmit its reading or not. If the difference in magnitude between u and another sensor whose reading is reported is less than a threshold, say δ, u will suppress its report. The suppression can also be done on a temporal scale. Sensors do not report their readings if there are no significant changes in their observed values over a period of time. When the sink receives reports, it can deduce the readings of sensors whose reports are not received based on the spatial and temporal correlation.

There are many subtleties in putting this seemingly simple suppression technique to work. Nodes can only overhear one hop neighbors directly. If we have a large step value, how can we suppress readings of sensors more than one hop away? How can we ensure that sensors with similar values in different regions transmit, that is, how can we make sure that iso-lines, that represent same values, do not become fragments in the sink? How can we reliably remove faulty sensors with outlier readings? With suppressed readings, how can the sink reconstruct the contour maps?

To enable a sensor u to suppress based on a report from sensor v within a given number of hops away, we stamp a packet containing reports with hop information. To implicitly coordinate who should report and who should not at what time, a sensor waits for a random time that is a function of its reading. In order to enable outlier detection, sensors observing a disparity in their local average will transmit even if they normally do not. This will assist the sink to detect outliers more accurately. In order to accurately reconstruct the contour maps, the sink utilizes its knowledge of the terrain of the sensor field and the location of the sensors for outlier detection and interpolation. Having done the outlier detection and interpolation, the sink applies a smoothing algorithm to reduce the errors introduced in previous steps.

Our contour data representation allows for trade-off between accuracy and communication cost of various granularity. It can have numerous applications. We focus on event monitoring and network health diagnosis applications. These applications include spatial-temporal event monitoring, residual energy monitoring and faulty sensor detection.

IV. Related Work

Hellerstein et al. [11] propose to construct isobar maps in sensor networks. In their work, each sensor is associated with a bounding box and an attribute. The isobar aggregates sensors with the same attribute that are close together. They show how in-network merging (either accurate or lossy) of isobars helps reducing the amount of communication. In the isobar computation, every sensor has to participate explicitly and pass merged isobars to its parent in the routing tree. Zhao et al. [25] applies similar techniques to construct an approximate view (at the sink) of the residual energy of sensors in the network. Further, Zhao et al. [26] propose a monitoring infrastructure that can identify system failures

and resource depletion. They propose a tree construction algorithms that enables energy-efficient computation of some classes of aggregates (sum, average, count) of network properties (loss rate, energy level, packet count, etc). In our work, we do not need every sensor to communicate (transmit), even with its neighbors, which saves energy and extends the lifetime of the network.

Bandyopadhyay and Coyle [3] address the issue of temporal and spatial sampling rates under conditions of minimum energy usage. However, they assume enough channels are available so that collision free scheduling can be constructed. Ganesan et al. [8] observe that spatio-temporal irregularities in sensor networks impact many performance issues in sensor networks. To mitigate the impact of irregularity, for data aggregation and compression, they propose to apply spatial interpolation of data and temporal signal segmentation followed by alignment. To reduce the cost for data-centric storing and routing, they propose to use virtualization and boundary detection. Our scheme automatically takes care of these irregularities since redundant reports from dense areas are suppressed and sensor readings from those with no direct report are interpolated spatially and temporally. A temporal prediction approach has been proposed in [29] that is similar to MPEG compression. In a more recent work [13], the authors study a node self-selection problem and compare several self-selection methods. The proposed solutions in both these work are similar in spirit to one component of our approach.

Many other data reduction techniques have been proposed in the literature. Some focus on computing simple functions over the readings of sensor networks, e.g., [15], [19], [24]. Madden et al. develop Tiny AGgregation (TAG), a generic aggregation service for sensor networks. TAG performs in-network aggregates by constructing a routing tree. Przydatek et al. [19] focus on the security aspects of data aggregation. Trading aggregation accuracy with communication cost is studied in [24]. Giridhar and Kumar [9] characterize the rate at which certain classes of functions can be computed and communicated in sensor networks. More sophisticated aggregation functions such as wavelet histograms has also been proposed [11]. Application-specific compression techniques are proposed in [21]. These sophisticated techniques work well with large amount of data produced by sensors. However, they are not suitable when the sensor data is small. Yoon and Shahabi [27] propose a CAG (clustered aggregation) technique to reduce communication overhead so that energy is saved. Ganesan et al. [7] propose a storage and search system for sensor network applications. They use in-network wavelet-based summarization and progressive aging of summaries to support long term querying in storage and communication-constrained sensor networks. They show that these mechanisms support efficient drill-down search over summaries. In addition to the aforementioned data reduction approaches, [31][32] analytically study how the spatial correlation of sensor data affects the efficacy of data compression and gathering. Compared to all these work, our algorithms target sensor networks with limited capabilities, i,e, sensors have limited storage space and computational power. Our proposed algorithms are light in terms of both computation and storage requirement. We believe that this is an advantage for an economical deployment of large-scale networks.

Observing that sensor networks are used in a wide variety of applications, some recent work [28] [30][33][34] start to characterize the sensor data being used. More specifically, Guestrin et al. [28] propose a distributed regression algorithm to do in-network parameter fitting for sensor data. After the algorithm is performed, the sensor data is modeled by a few parameters and thus queries can more efficiently replied. Rossi et al. [30] describe another distributed algorithm for estimating parameters of sensor data. Nevertheless, their algorithms are more suitable for diffusion phenomena. The authors of [33][34] show that many realistic sensor data exhibit spatial correlation. They also provide methods to generate realistic sensor data with varying degree of spatial correlation. When we evaluate the performance of our proposed algorithms, we use the method of [33] to generate sensor data with different spatial correlation.

V. Algorithms for Contour Map Construction

In this section, we describe the basic operations of the proposed algorithms. More specifically, we describe the spatial suppression algorithm at individual nodes and the interpolation and smoothing algorithm at the sink. We will leave other optimization to Section VI.

A. Local Suppression at Each Sensor

A report in a sensor can be triggered by either sensing an event, or the expiry of a periodic timer, or by a query from the sink. In these instances, there might be spatial or temporal correlation between sensors' values which can be used to suppress the reports at a particular sensor.

1) Spatial Suppression: When a sensor node is triggered to send a report by an event , it is very likely that the same event triggers the neighboring nodes. In order to avoid redundant transmissions, we can choose those nodes with more relevant information to transmit while suppress other nodes with less relevant information. More precisely, we assume that a message with a larger reading value is more relevant, we then let each sensor u back off for a time period with the length randomly chosen from the range $[0, 1/M_e^u]$. Here M_e^u denotes the magnitude of reading triggered by an event e at node u.

We further use $N(v)$ to denote all the sensors within the communication range of node u. Once node u overhears some nodes in $N(v)$ transmit their readings, u averages over all these overheard readings to get M_e^{avg}. Node u then compares M_e^{avg} with M_e^u. If the difference is less than a constant δ, u suppresses its report. Otherwise u will send its own reading. The pseudo-code for this entire process is presented in Figure 1.

```
On sensing an event e at sensor u with magnitude M_e^u
1.  random_wait(1/M_e^u)
2.  if (heard sensor v ∈ N(u) reporting M_e^v)
3.     compute average overheard magnitude M_e^avg
4.     if ( | M_e^avg − M_e^u |≤ δ)
5.           then do not transmit and return
6.  transmit when timer expires
```

Fig. 1. Event processing at sensor u

It is important that we compute the average M_e^{avg} and suppress based on it. We illustrate this subtlety using the following example.

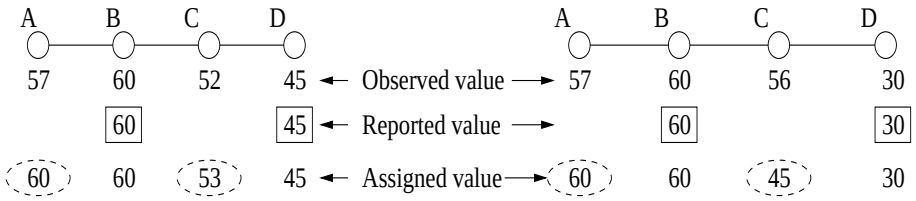


Fig. 2. Spatial suppression example

In the left scenario depicted by Figure 2, node B observes a reading of 60, while its neighbors, A and C, observe readings of 57 and 52 respectively. If B transmits (reports) its value first, then, assuming $\delta = 10$, both A and C will suppress their reports. D observes a reading of 45, and even if it transmits, C's reading is within $\delta = 10$ of D's reading, and hence C will remain quiet.

An alternate situation occurs when C reports a reading of 56 and D reports a reading of 30, as shown by the right scenario in Figure 2. Such a situation can easily occur when there is a physical obstruction that prevents the event registering at sensor D, or there are other physical parameters influencing the readings. If B and D transmit, then C is in a dilemma, because only one of its neighbors' values is correlated with its own reading. If the sink took a simple average of C's neighbors to compute C's perceived value, then C will be assigned a value of 45, which is more than $\delta = 10$ away from the actual reading at C. To reduce the interpolation error in the sink or to better detect outlier readings, when C overhears B and D, it computes the average of these two readings, and if the average is within δ of its actual reading, it suppresses its event report.

For sensors that are on the routing path, the spatial suppression process works in the same way as for non-relay nodes, but with one distinction. Since relay nodes have to transmit to convey messages from other nodes, they can potentially piggyback their own sensor readings. The advantage is that more information is sent to the sink without media access overhead. The disadvantage occurs when the report size generated at each sensor is large, resulting in significant packet transmission times when more and more reports are added to a packet. One way to limit this negative impact is to limit the number of reports carried in a single physical packet. The size of the cumulative packet should not exceed the maximum physical packet size allowed by the link layer.

One of the salient features of the spatial suppression scheme is that it does not depend on how packets are routed in the network. In other words, it is independent of the routing protocol. We do not rely on any explicit or implicit

constructions of the routing tree as in [14]. Routing path information can be conveyed to the sink, if needed, using path digests as discussed in Section VI.

2) Temporal Suppression: We define two parameters for each sensor: a minimum reporting interval τ_{min}, and a maximum reporting interval τ_{max}. Successive reports from a sensor must be spaced at least τ_{min} seconds apart, while each sensor must report at least once every τ_{max} seconds. The value of τ_{min} is dependent on the type of monitoring application, and is chosen to capture all event occurrences in the network. Typically, $\tau_{max} = k.\tau_{min}$, where k is a large number. If a report is not received from a sensor for $J * \tau_{max}$ seconds where J is a small number, e.g. 3, then the sink will assume that the sensor is dead.

Define epochs as time periods of length τ_{min}. For a given sensor v, Let $M_e^v(t)$ and $M_e^v(t-1)$ be the samples in epoch number t and $t-1$ respectively. We exponentially average the readings with a low pass filter with filter constant, x. i.e. $M = x.M(t) + (1-x).M(t-1)$. Note that we can also use the last m samples, as opposed to using the last sample alone, for the averaging process. The sensor reports $M(t)$ to the sink if either $\mid M - M(t) \mid > \delta$ or time elapsed since last report is greater than $\tau_{max} - \tau_{min}$.

The temporal suppression phase introduces a potential conflict with the spatial suppression phase, where one phase wants to suppress a event report while the other phase wants to transmit the report. We resolve this by suppressing a sensor report only when both the spatial and temporal phases request to suppress. Spatial suppression is based on sensor readings most recently heard from neighbors.

B. Interpolation and Smoothing at the Sink

When the sink receives a subset of sensors' readings, it will first perform interpolation and assign readings to sensors without a direct report. It then performs a smoothing procedure.[1] This procedure is necessary because the readings after the interpolation can vary over a large range, depending on δ. The actual readings are, in general, much more smooth, spatially and temporally. This procedure can reduce errors significantly. The interpolation step can not be avoided since it assigns a value to a sensor that is within a certain range imposed by the suppression criteria. Without this step, the smoothing procedure can assign a value that is well outside of the range. We describe these two algorithms in detail below.

1) Interpolation Algorithm: The goal of interpolation is to assign values to those sensors that do not report their actual readings due to the suppression. We assume the data sink knows all the routing path information before the interpolation. This is achieved by a Bloom Filter technique, which will be described in Section VI-A.1. We refer those sensors for which readings are reported as *black nodes*, and sensors for which sensor readings are unknown as *white nodes*. We define the level of interpolation, $I = x$, if the interpolation process assigns values to white nodes that are at a minimum distance of x hops from a black node. The maximum level of interpolation required for a given event is defined as I_{max}. If we assume no reports are lost in their delivery process, we should have $I_{max} \leq 1$ given the above suppression mechanism. However, reports can be lost due to congestion or channel errors. Such lost reports may require a value of $I_{max} > 1$. On the other hand, T_{max} should have a upper limit so that large interpolation errors can be avoided.

We now use the pseudo-code in Figure 3 to describe the procedure of constructing event contours.

```
After receiving event reports at the sink,
1. Assign signal strengths M_e^u for black nodes
2. I = 1
3. while I < I_max
4.    for any white node v within I hops from black node
5.          M_e^v ← average values of neighboring black nodes
6.          color(v) = gray
7.    Change all gray nodes to black
8.   I ← I + 1
```

Fig. 3. Contour reconstruction: interpolation at sink

[1]Smoothing is done over sensors that are in the same physical region. This can be done easily at the sink, since it has all the geography knowledge.

If $I_p = 0$, the only information we have is the received reports from the black nodes. If $I_p = 1$, then we have interpolated data from most of the white nodes, i.e., those that have suppressed their reports. The exception is when two neighboring nodes have sent reports and one of the two reports is lost. In this case, we will have incorrect interpolation, but bounded by the difference of the node's reading from the average of the neighbors' readings. If $I_p = 2$, nodes whose readings are interpolated at this stage are either (a) nodes whose reports are lost or (b) nodes whose neighbor's reports have been lost and interpolated in the previous stage ($I_p = 1$), (c) nodes that have not sensed an event at all. The case $I_p \geq 3$ will be reached only when the loss rates are pretty high, or they have not sensed the event at all.

In order to reduce errors in the spatial interpolation process, we set $I_p = 2$. This limits our exposure to packet losses and can help the sink in figuring out the regions that experience losses due to errors and/or congestion. In addition to spatial interpolation, the sink uses the previous values from a given white node to assign sensor values, using temporal interpolation. For each given white node v, the sink computes the spatial interpolated value, and uses the previous value assigned to v, and compares the difference of these two values to the current event reports received in the neighborhood. The sink uses that value which has the smaller variance when compared to the reports from neighboring black nodes, as the chosen value for that particular sensor v.

2) Smoothing the Constructed Contour: In the previous interpolation algorithm, any sensor that does not have its actual reading received by the sink is assigned a reading value from the average of its nearest neighboring sensors. This local decision may cause the contour to be sharply discontinuous at certain sensors, which does not fit the reality in most cases. To address this issue, we use a circular Gaussian kernel to convolve the interpolated reading values. This way, a sensor with unknown reading value is assigned a distance-weighted sum of its nearby reading values. Hence, the reliability of the interpolation can be improved, especially when the sensor network data are spatially correlated. Note that we can not perform smoothing without interpolation. The interpolation process recovers important information about the range of sensor values. Without interpolation, our smoothing technique can assign values that are outside the range that are imposed by the suppression algorithm.

To be more precise, the circular Gaussian kernel function is expressed as $G(r) = ne^{-\frac{r^2}{2\sigma^2}}$, where r is the distance measured by hop number, n is a normalization factor, and σ controls how the importance of nearby reading values degrades over r. Suppose L_r represents the set of nodes r-hop away from node v, then after smoothing, the reading value for node v, which is denoted by M^v, should be

$$M^v = \sum_{r=1}^{3} \sum_{u \in L_r} M^u n e^{-\frac{r^2}{2\sigma^2}}$$

We only consider those nodes within 3-hop distance into the smoothing because we have set $I_p = 2$ in the previous interpolation procedure. Therefore, a sensor node without an actual known reading value should have at least 2 neighbors within 3 hops. σ is an important factor in the above operation. Intuitively, a large value of σ gives more weights to nodes closer to v, and this will work best for sensor data with lower degree of spatial correlation. On the other hand, a small value of σ will average the reading values within a large neighboring area, and thus works for sensor data with high spatial correlation.

C. Discussions

In our previous discussion, the local suppression algorithm is performed by individual sensors, while the interpolation and smoothing algorithm is used by the sink. These two algorithms work in concert to reconstruct a contour map for the entire sensor field. Regarding the correctness of this scheme, we can easily have the following two remarks.

Remark 1: If we assume no packet is lost due to collision or other types of communication failures, then the local suppression algorithm ensures that any node observe at least one transmission within its 2-hop neighborhood. In other words, the range spanned by suppressed nodes are bounded, thus the sink can collect samples from every area in the sensor field.

Remark 2: We assume no packet lost and the local suppression algorithm works correctly. By setting $I_p = 2$, the interpolation and smoothing algorithm will assign a value to every suppressed node, and the difference between the assigned and actual value is less than δ.

In practice, packets may be lost due to collision, channel error or other factors. Accordingly, the above two observations may not hold. To address this issue, we can increase I_p in the interpolation algorithm to more aggressively assign values to nodes whose reading values are not available to the sink.

VI. Extensions

We discuss several important extensions in this section. We show how we enable multihop local suppression, and how data aggregation is done with local suppression.

A. Multihop Local Suppression

To enable more aggressive tradeoff between accuracy and communication cost, we propose techniques that enable suppression that are based on overhearing of sensors that are multiple hops away. Given an upper bound h on the number of hops, if a sensor u overhears a report that is within h hops of itself (indicated in the report by a field indicating the number of hops traveled), and if the report has a value that is within δ of u, u will suppress its transmission. In order for the sink to correctly interpolate the values of sensors whose report are not received, we utilize the routing information. In the following, we first describe an efficient mechanism that keeps the sink up-to-date about the routing information, we then briefly describe the interpolation algorithm at the sink.

1) Routing Information: The sink needs to know the routing tree which consists of all the routes from data sources to the sink. If the routing tree does not change, we can attach this information in the packet header. However, routing paths are maintained in a distributed manner, and they may change as nodes die or go to hibernating mode. Therefore, a more efficient mechanism is required for the sink to acquire the route information. In the following, we propose to use a Bloom filter technique to carry the route information.

The sink first creates a number of hash functions and broadcasts these information to every data source. Once a data source generates a data packet, it allocates a bit array in the packet header to be used as a Bloom filter (details about Bloom Filter can be found in [4]). The bit array is initialized to zero. Now if the packet traverses a sensor node (including the data source itself), the node uses the created hash functions to hash its node ID into some bits of the bit array. These corresponding bits are set to one. Eventually, when the sink receives the packet, the sink can apply the same set of hash functions to each node ID u (we assume the sink knows the set of node IDs). If all the bits in the indices of the hash values are one, the sink infers that the node u is very likely on the route. Such an inference might bring errors because of false negative (a node on the route is missing from the inference) or false positive (a node not on the route is inferred to be on the route) . Since the Bloom filter technique does not have false negative, the sink only needs to deal with false positive. For this purpose, once the sink determines the potential candidate nodes set S, it will try to piece together the routes from the sink back to the data source (reverse route) by using the geographical information for all the sensors. For instance, if a node is a potential candidate for a route but it is far away from the data source, the sink will treat the node as a false positive and removes it from the candidate list. In practice, we can select the number of hash functions to reduce false positive probability, based on the expected number of deployed sensor nodes. Such a mechanism is lightweight and deals with false positives very well based on location information. For example, for path length of 20 hops, we can use a 4-byte Bloom filter array and one hash function with a false positive probability for one hop of $f = (1 - e^{-20/32}) = 0.46$. While this is very large, a false positive path of 5 hops has a probability of $f^5 = 0.02$, and the probability for a false positive path of 10 hops drop to $4.7 * 10^{-4}$. Therefore, with only 4 bytes in the packet, we can eliminate all false positive paths with high probability.

2) Interpolation Algorithm at the Sink: When the sink receives the reports from the sensors, it iteratively performs interpolation to obtain the values of the sensors that are within i hops of at least one sensor that has reported value directly where $1 < i \leq h$. For $i = 1$, the algorithm is what we have described before in Section V. All we need to know is that whether they are direct neighbors of each other. For $i > 1$, we need to know the hop distance between two sensors in the routing path (or tree). If two sensors are within i hops, then the one that did not report is within δ of the sensor who directly reported its value, if the reporting sensor is farther away from the sink than the non-reporting sensor.

B. Interaction with Data Aggregation

Certain monitoring tasks require the computation of a function on the sensor data, e.g. max, min, average of energy level of sensors in a sensor network. Due to the simplicity of these functions, they can be computed in network. In

some sensor networks, there are special nodes called aggregators that perform such function. In other sensor networks, every sensor can be an aggregator.

Our framework can accommodate function like max, min with the idempotent property, namely, the function value is the same by repeatedly applying the function to the same input. For example, to compute the max in network, the proposed local suppression algorithm can be modified so that a sensor suppresses its own report whenever it overhears a larger value.

To accommodate non-idempotent functions like average and sum, the aggregator needs to know the routing information. The proposed Bloom filter approach can be readily applied to serve this purpose. Since an aggregator needs to collect reports from every sensor. Our proposed interpolation algorithm can recover the reading values for those sensors which do not send packets to the sink directly. The algorithm is almost the same as we described previously. The aggregator computes the function value on the interpolated readings of all sensors. If routing spans all sensors and each non-leaf node is an aggregator, each non-leaf node needs to know its two-hop neighbor information in order to correctly perform interpolation. The function value computed in this way is an approximation to the actual value. In many situations, this is still worthy because a large chunk of communication overhead are saved.

VII. Application Scenarios

In this section, we describe three key applications of our contour mapping algorithm.

A. Spatio-Temporal Event Monitoring

The set of events monitored by a sensor network belongs to diverse categories. Events can be stationary or mobile events. In this paper, we primarily consider stationary diffusion events, where the event source does not move and the magnitude of the observed event decreases with distance from the event source. However, our proposed contour construction algorithms are easily applicable to other types of events. Various physical processes, which sensors are designed to monitor, are typically modeled as diffusion events with varying dispersion functions. The frequency of these events can vary widely and multiple events may occur at the same time.

Physical events with diffusion spread processes are principal events targeted for monitoring using sensor networks. For example, temperature, air quality, chemical sensors follow a diffusion spread process, i.e., the samples are highly correlated over any small region. In addition, such events are very much likely to vary over time.

B. Residual Energy Monitoring

Diagnosis of the health of a sensor network is a critical operational requirement. An important parameter of sensor health is the residual energy level of each sensor since sensors are severely energy-constrained. Keeping tabs on residual energy allows dynamic reconfiguration of the network for optimal performance.

Zhao et al. [25] discusses a polygon-based mapping approach to aggregate data for gathering residual energy information on sensors. However, all sensors are required to transmit their residual energy information, including all nodes that have low energy levels. In addition, each sensor needs to compute the merging of polygons which requires more computational power. There are several benefits when our contour mapping approach is applied to monitor residual energy levels of sensors. Consider two scenarios: (a) one node has very low energy level in a neighborhood, and (b) an entire neighborhood has low residual energy. In case of (a), it is important to identify which other nodes have better energy levels. When the low energy node sends out its energy report, local suppression will immediately trigger nearby nodes with higher energy levels to transmit their energy reports, giving the sink alternate information right away. In case of (b), it is not necessary for all nodes to send their low-energy level reports as it is a waste of energy. Local suppression will ensure that only a small subset of nodes send their energy reports and be able to convey the requisite information at the same time.[2]

Instead of using a single step value function for the contour mapping, we use a magnitude dependent step function $\delta(E^v)$, where E^v is the residual energy level at sensor v. We assume that each sensor starts out with maximum energy

[2]Instead of one node and all nodes in a neighborhood, the discussion is valid when replaced with a small subset and a large subset of nodes in a neighborhood, respectively.

E_{max}, and is considered dead if the residual energy falls below E_{min}. Now, we use a binary exponential step function to define different thresholds, i.e.,

$$\delta(E^v) = \frac{E_{max}}{2^{m+1}}, \text{where } E_{min} \leq \frac{E_{max}}{2^{m+1}} \leq E^v < \frac{E_{max}}{2^m} \quad (1)$$

for the smallest positive integer m. The reasoning behind this is that nodes with high energy levels do not need to report their status frequently.

The energy report can be queried periodically by the sink from all the sensors in the network or it can be triggered when energy levels fall below certain thresholds such as two or three step-levels above E_{min}. The relevance of a residual energy report is more for nodes with low energy levels. In other words, while reporting residual energy, nodes with high energy backoff more than nodes with low energy.

C. Faulty Sensor Detection

Another aspect of sensor health monitoring is to detect the existence of faulty sensors. Faulty sensors are those nodes which report readings sharply inconsistent with the actually observed event values at the nodes. In certain scenarios, faulty sensors can be caused by obstacles that significantly distort the readings. Note that faulty sensors can be detected only when the value they send out is different from the observed value of the event. We detect faulty sensors in two steps.

In the first step, we apply the contour mapping algorithm to the sensor field, as we did in regular scenarios. Since each sensor computes the average of all reports originating at the neighbors (the black nodes) and then decides to suppress based on that value, it follows that any sensor that reports faulty readings that are significantly different from the neighboring sensors will have its neighbors also transmit their reports. Thus, any potential faulty sensor(s) reports will be accompanied by more neighbor reports than usual, which helps us pinpoint any clusters with faulty sensors.

In the second step, we use *outlier detection algorithms* to help pinpoint potential data which stand out from the rest of the dataset. A variety of outlier detection algorithms exist in the literature. The difference between these various algorithms is primarily on the implementation complexity. We use the outlier detection algorithm in [17] to detect the outliers in the contour data, and label them as faulty sensors.

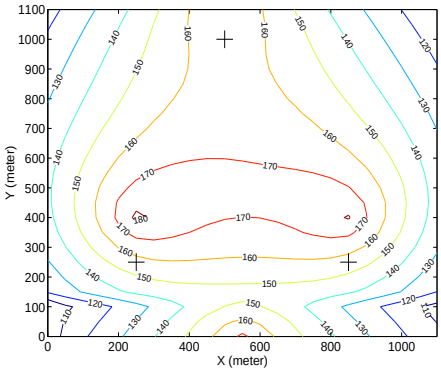
VIII. Performance evaluation

We use various scenarios to evaluate multiple aspects of the proposed algorithms. These include the basic spatial and temporal suppression for sensor nodes, the contour construction algorithm at the sink, and combining the basic design with outlier detection algorithms to identify faulty sensors, if any. In our simulations, we use three sensor datasets: a temperature diffusion process caused by fire sources, a dataset with high spatial correlation and a dataset with low spatial correlation. We also evaluate an important application of the proposed design, that is, monitoring residual energy in sensor networks.

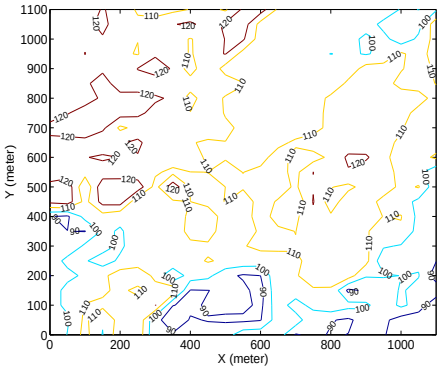
A. Experimental Settings

Our simulation platform is *ns-2* [16]. The assumed physical model is *TR3100* transceiver [18], manufactured by RF Monolithics [3][20]. This transceiver can work in either a high power mode or a low power mode. In the high power mode, the transmission rate is 576 kbps and can be used for data delivery. The transmission and reception power requirements are $30mW$ and $21mW$ respectively. In the low-power mode, the data rate is 19.2 kbps and used for control message exchange. The power consumption is 90% lower than in the high-power mode. To fully utilize such a dual-channel interface, we adopt the energy management scheme proposed by [5]. In such a scheme, a node turns off its data-channel when it has no traffic, yet its control radio is still on so that the node knows when other nodes need it to forward traffic[4]. The transmission range for each sensor is 100 meters.

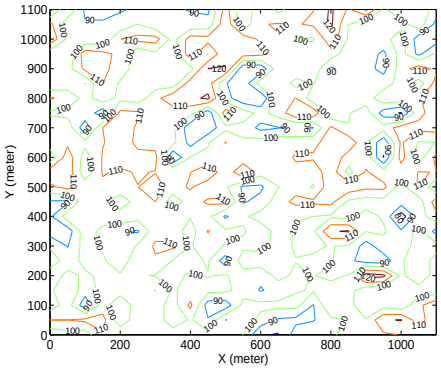
The MAC protocol used here is CSMA/CA. We assume that sensor nodes know the location of the sink and use a simple geographical routing protocol similar to LAR [22]. While this simple protocol does not produce the best delivery ratios possible, we use it as a baseline case. A better routing protocol will provide better performance with our algorithms.



4.1: Temperature diffusion

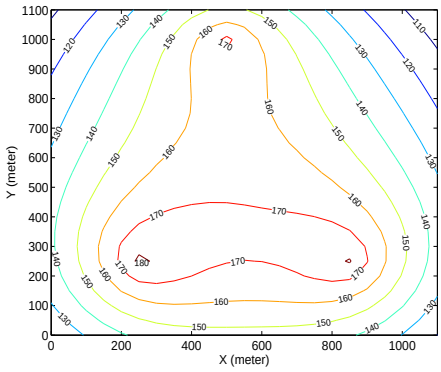


4.2: A dataset with low spatial correlation

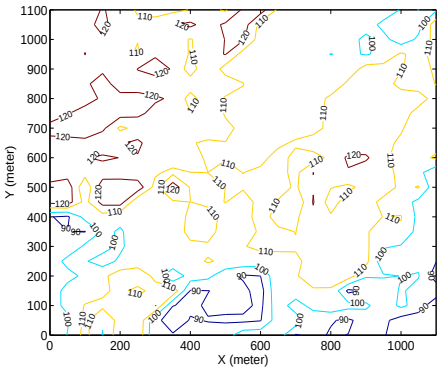


4.3: A dataset with high spatial correlation

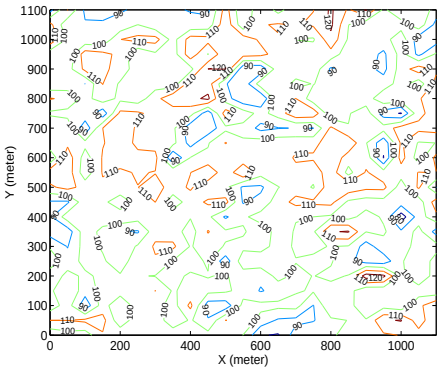
Fig. 4. Contour maps for the three datasets used



5.1: Temperature diffusion

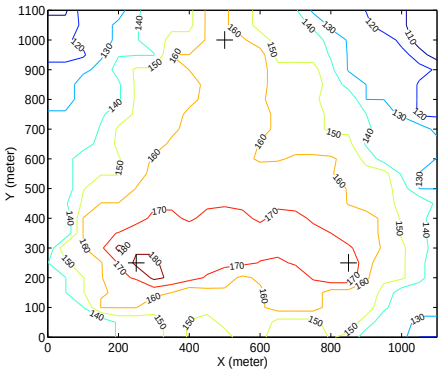


5.2: A dataset with low spatial correlation

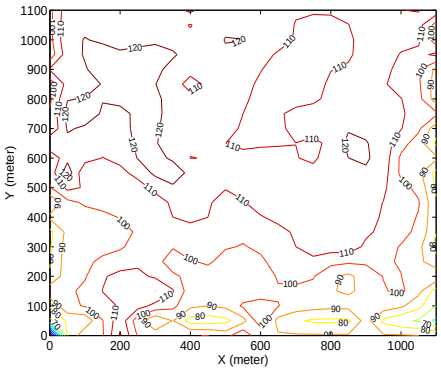


5.3: A dataset with high spatial correlation

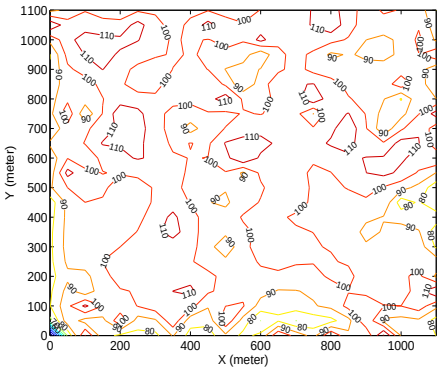
Fig. 5. Recovered contours by using all reports sent out (by assuming packet loss ratios are zero)



6.1: Temperature diffusion

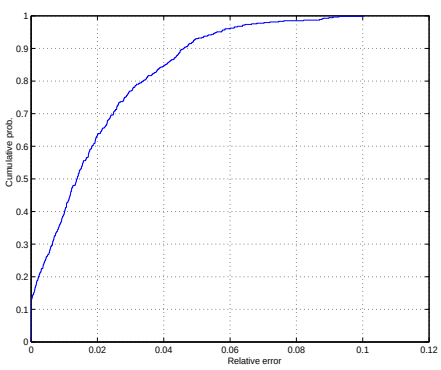


6.2: A dataset with low spatial correlation

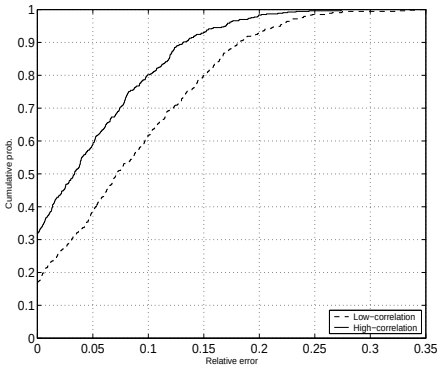


6.3: A dataset with high spatial correlation

Fig. 6. Actual recovered contours (Packet loss ratios are 18.4%, 16.3%, 14.6% respectively))

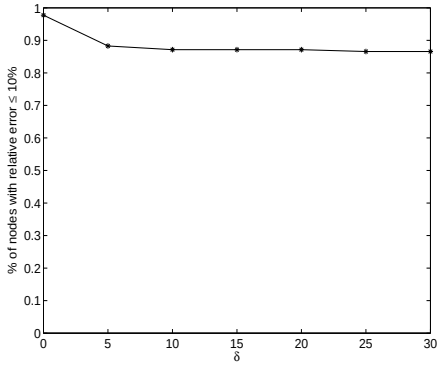


7.1: Temperature diffusion

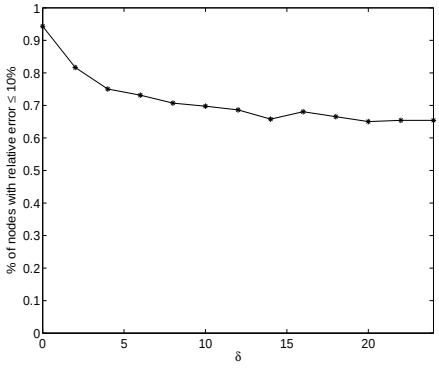


7.2: Datasets with high and low spatial correlation

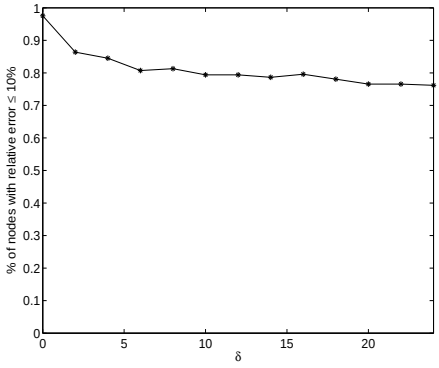
Fig. 7. Cumulative distributions of the relative errors at sensor nodes



8.1: Temperature diffusion

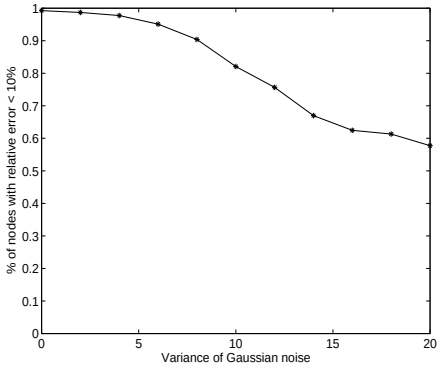


8.2: A dataset with low spatial correlation

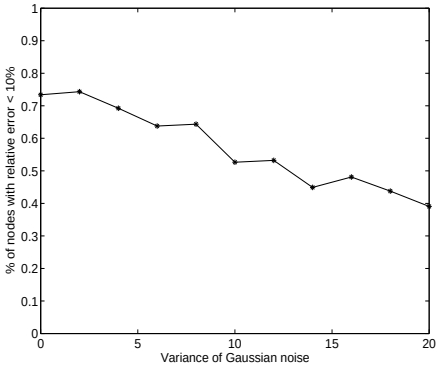


8.3: A dataset with high spatial correlation

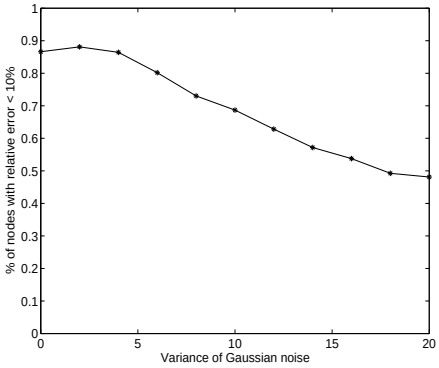
Fig. 8. Accuracy of reconstructed contours vs. different δ



9.1: Temperature diffusion



9.2: A dataset with low spatial correlation



9.3: A dataset with high spatial correlation

Fig. 9. Accuracy of reconstructed contours vs. Gaussian noise

The network topology is a 23×23 grid. The distance between neighboring sensors is 50 meters. The data sink is at the left-bottom point of the topology. Given such a network topology, the source and diffusion model, we can generate reading values for each sensor node. We focus on the grid topology in this paper in order to bring out all the aspects of the contour mapping scheme. We have tested our scheme with random topologies, and the results presented here are representative of results with random topologies.

In our experiment, we used three types of sensor data. The first sensor data simulate the temperature diffusion from fire sources. In such a diffusion model [1], a point p in the space has a value which is the summation of diffusions from multiple point sources. This summation has the following expression:

$$V(p,t) = \sum_{i=1}^{N_s} [k * dist(i) + 1]^{-a} * M(i,t)$$

where $V(p,t)$ is the value at point p at time t. N_s is the number of event sources. $M(i,t)$ is the value for source i at time t. $dist(i)$ is the distance between p and the source i. a, k are constants, chosen to be 3 and 0.05 respectively. We simulate a scenario in which three point sources are located at $(250, 250)$,$(500, 1000)$,$(850, 250)$ (the unit is in meters) and their $M(i,t)$ are 145, 182, 160 respectively. Given the network topology and fire sources are fixed, each sensor node can easily compute the value it should observe. The actual contour is depicted in Figure 4.1 in which the three $''+''$ represent the fire sources.

The second and third sensor data we used are based on a recent paper [33] which provides methods to produce realistic sensor datasets with varying spatial correlation. To be representative, we used such methods to create two datasets: one with low spatial correlation while the other has high spatial correlation. The contour maps for these two datasets are depicted in Figure 4.2 and 4.3. By comparing these two figures, we can clearly observe that fewer contours are needed for representing the dataset with low spatial correlation.

Without explicitly mention, we assume $M(i,t)$ is time-invariant, thus the actual reading values for each sensor node is constant over time. An example using both spatial and temporal variant events will be presented later in this section. In time-invariant scenarios, we assume each sensor node is triggered by the sensing task once. Note that we only employ the single-hop suppression algorithm, which is described in Section V. The performance for multi-hop suppression algorithm is part of future work.

B. Performance of Contour Construction Algorithm

In this section, we first evaluate the quality of the constructed contour maps. We then study how the quality is affected by factors such as the choice of δ and measurement noise.

1) Quality of reconstructed contour maps: We reconstruct contour maps for the three datasets by setting $\delta = 10$ and not imposing any measurement noise. For the temperature diffusion dataset, 92 out of the total 528 sensors actually sent out their reading values while 75 readings are successfully received by the sink. In other words, 17 readings are lost due to communication failures. For the second dataset (low spatial correlation), 92 sensors sent their readings while 77 of them are received. For the third dataset (high spatial correlation), 123 sensors sent their readings while 105 of them are received by the sink.

We first show the reconstructed contour maps in Figure 5 by using all those readings that are actually sent out. This equals to assume no packet loss. By comparing Figure 4 and 5, we can notice that the reconstructed contour maps are quite consistent with the original ones. This demonstrates the efficacy of the proposed algorithms. Next, we show in Figure 6 the reconstructed contour maps by using the actually received reports by the sink. The contour maps in Figure 6 deviates from their original shapes to some extend. Nevertheless, such a deviation is largely caused by the limited capability of the underlying wireless communications.

In addition to visually comparing the contour maps, we present the cumulative distributions for the nodes whose relative errors are less than 10%. Such distributions are shown in Figure 7. Figure 7.1 is for using the temperature diffusion data. In this scenario, 90% sensor nodes have a relative error less than 5%. Figure 7.2 compares the accuracy when using datasets with high and low spatial correlation. Figure 7.2 shows that about 80% sensor nodes have relative

[3]The predecessor of *TR3100* is *TR1000*, used by the UC Berkeley Motes [2].

[4]We can additionally use a MAC protocol that periodically wakes up to check for events, reports or neighboring transmissions [23], in order to garner extra savings in power, however, it is beyond the scope of this paper.

	Dataset 1			Dataset 2			Dataset 3		
	transmit	overhear	receive	transmit	overhear	receive	transmit	overhear	receive
No suppression	1.8	77.4	1.1	1.4	73.4	1.0	2.4	107.4	1.7
Proposed algorithms	0.2	8.9	0.1	0.2	8.7	0.1	0.3	12.5	0.1
Energy saving	88.2%	88.5%	88.4%	85.7%	88.1%	90.0%	87.5%	88.4%	94.1%

TABLE I

ENERGY SAVING PER SENSOR NODE (ENERGY IS MEASURED BY MJ)

errors less than 10% in the high correlation scenario while only 62% nodes have relation errors less than 10% in the low correlation scenario. This indicates that the constructed contour is more accurate when the sensor data exhibits higher spatial correlation. This is understandable since higher spatial correlation means that the measurement values at neighboring nodes have less difference, thus suppressing or losing a few reading values does not significantly affect the interpolation accuracy.

2) Impact of δ: The step-value δ is an important parameter for the local suppression algorithm. We study how different choices of δ affect the fidelity of reconstructed contour maps. To this end, we define a metric: *the percentage of nodes whose values are within 10% of their original ones*. We use this metric to measure the accuracy of a contour map. A larger value of this metric means that the reconstructed contour map has higher fidelity. We then show how this metric changes over different choices of δ in Figure 8. For all the three datasets we used, the accuracy measured by the above metric remains stable when $\delta > 20$. This is because we apply suppression to a single hop neighborhood in which the reading values do not differ much. As a result, when δ is larger than a certain threshold, there is always one node sending out reports. If we use a multi-hop suppression mechanism described in Section VI-A, the accuracy will decrease with δ, because the number of nodes that send reports will be severely limited. In Figure 8, we only show the results for $\delta \in [0, 20]$. A general observation from the figures is that smaller δ will increase the accuracy of the reconstructed contour maps. However, the price is to increase communication overhead as well as energy consumption.

3) Sensitivity to noise: In this scenario, we evaluate how measurement noise reduces the quality of the reconstructed contour maps. To this end, we assume the measurement at each sensor involves a Gaussian noise with zero mean and a standard deviation σ. We then vary σ and measure the accuracy of the constructed contour, which is still measured by the *percentage of nodes having relative error* $\leq 10\%$. For all the different σ, we fix δ to be 10. The results are shown in Figure 9. We can see that the accuracy is not reduced much when σ is less than 5, which is exactly a half of the δ being used. Nevertheless, when σ becomes comparable or even larger than δ, the accuracy degrades significantly. Our explanation is that at large noise level, the neighboring reading values can easily differ more than δ, which makes the suppression mechanism ineffective. As a result, large number of transmissions occurs and serious MAC layer contention leads to heavy packet loss.

4) Energy saving: We evaluate the energy consumption of the proposed mechanism by comparing it to the scenario where the local suppression algorithm is disabled and thus all nodes transmit their readings. We classify energy consumption into three categories: energy needed to transmit readings, energy needed to overhear neighbors' transmissions, and energy needed to receive transmissions meant for the sensor. We compute the average energy consumed over all sensors and summarize the results in Table I. Energy for overhearing neighbors turns out to contribute most to the overall energy consumption, as expected in sensor networks. In each component, the energy savings using the contour mapping scheme is up to an order of magnitude.

C. Detection of Faulty Sensors

One application for the contour mapping scheme is to detect faulty sensors. The mechanism for detecting faulty nodes is described in Section VII-C. To evaluate this aspect, we use a simplified version of the LOCI algorithm [17] to detect faulty sensors by using the temperature diffusion dataset. We randomly choose m sensor nodes and assign each of them a random value which deviates more than 3δ from the actual value at that node. While at the sink, we calculate the variance of each sensor's value from its one-hop neighbors to detect the faulty nodes. Figure 10 plots the variance for a scenario with $m = 2$. The two peaks in Figure 10 are located at $(250, 250)$, $(750, 400)$ respectively, correctly indicating the location for the two faulty nodes.

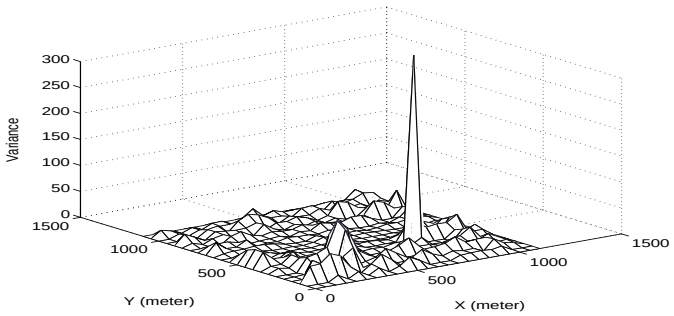


Fig. 10. Variance of reading values at sensor nodes - the faulty sensors are at $(250, 250)$ and $(750, 400)$

$[0, 200)$	$[200, 400)$	$[400, 600)$	$[600, 800)$	$[800, 1000)$
351	328	311	302	242
80%	82%	83%	82%	91%

TABLE II

RESULTS FOR TEMPORAL SUPPRESSION (FIRST ROW: TIME INTERVALS. SECOND ROW: TOTAL REPORTS SENT. THIRD ROW: PERCENTAGE OF REPORTS RECEIVED BY SINK.)

D. Spatial-temporal Event Monitoring

To evaluate the performance of the combined spatial-temporal suppression algorithm, we simulate a time-varying event in the context of using the temperature diffusion dataset. Specifically, we let $M(i,t)(i = 1, \ldots, 3)$ increase linearly over time, i.e., $M(i,t) = M(i,0) + \alpha_i t$. The three sources have various increasing rates, $\alpha_1 = 0.05, \alpha_2 = 0.07, \alpha_3 = 0.1$. Consequently, the actual reading value at sensor nodes may increase at different rates. We simulate a 1000-second time period during which each sensor node expects to send the observed value for every 200 seconds.

Without temporal and spatial suppression, there should be $5 * 528 = 2640$ reports generated. We enable both temporal and spatial suppression. Both suppression mechanisms have the same threshold, $\delta = 20$. Table II gives the number of reports actually sent out and received for each 200-second time interval. We make two observations: (1) the total reports sent within each time interval is much more than in the previous time-invariant scenario. This is because we are simulating an asynchronous scenario in the sense that sensor nodes check their values at different time instance. For a sensor node, its reading value which was suppressed earlier may no longer be suppressed since the value is changing over time. (2) The total number of reports generated in each time interval decreases over time. This is due to the temporal suppression. Accordingly, the packet delivery ratio improves tremendously from 80% in the first epoch to 91% in the fifth epoch. In overall, 40% reports are suppressed due to both spatial and temporal suppression.

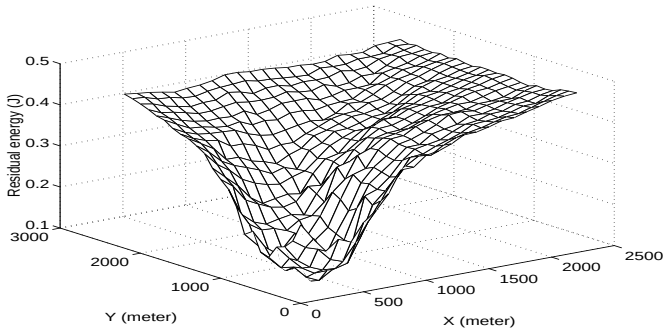
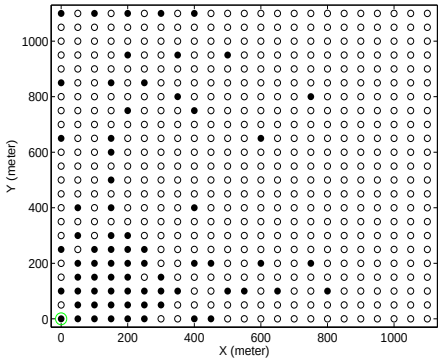
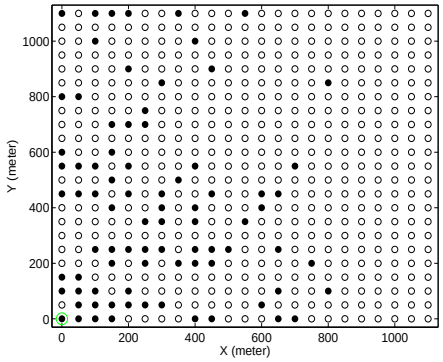


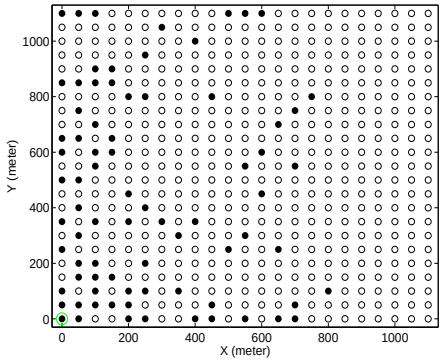
Fig. 11. Contour of residual energy in sensor nodes



12.1: Temperature diffusion (71 nodes send reports)

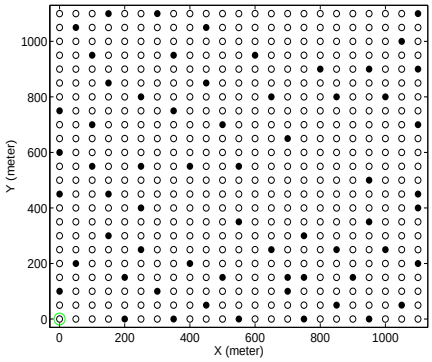


12.2: A dataset with low spatial correlation (82 nodes send reports)

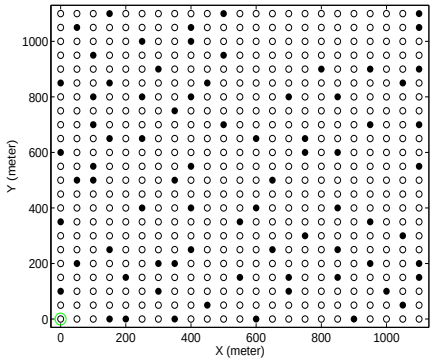


12.3: A dataset with high spatial correlation (78 nodes send reports)

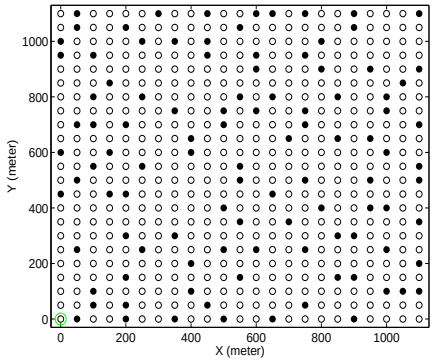
Fig. 12. Sensor nodes that actually sent out reports by using CAG ($\tau = 0.05$)



13.1: Temperature diffusion (62 nodes send reports)



13.2: A dataset with low spatial correlation (76 nodes send reports)



13.3: A dataset with high spatial correlation (104 nodes send reports)

Fig. 13. Sensor nodes that actually sent out reports by using our local suppression algorithm (sink is located at the left-bottom corner)

E. Contour Maps for Residual Energy Monitoring

Another application of contour mapping is to diagnose the health of the network by monitoring residual energy in the sensor network. Since energy monitoring is usually used on large time scale, we simulate a 7000-second data collection process in which each sensor generates a report for every 1000 seconds. The initial energy for each sensor node is assumed to be uniformly distributed between 0.2 and 0.5 Joules, since all sensor nodes may have delivered uneven volumes of traffi c.

The sink collects residual energy reports on a periodic basis from sensor nodes. Nodes send reports according to the spatial-temporal suppression algorithm and the step function described in Section VII-B. We subdivide the levels obtained in Equation 1 into two to obtain a total of four energy levels.

We plot the constructed residual energy contour at T=1000 seconds in Figure 11. It shows the residual energy contour built by the proposed algorithm. It can be seen that the nodes closer to the sink consume much more energy, which is to be expected in sensor networks since nodes close to the sink have to route packets for the rest of the network, and this routing load increases as the distance from the sink decreases. Out of the 529 nodes in the sensor network, the contour is drawn based on reports from only 116 nodes, yet the error in measurements is less than 10%.

F. Comparison with CAG

Our proposed algorithms share lots of similarity with CAG [27]. CAG is a clustered in-network aggregation scheme particularly suitable for sensor networks with spatially correlated sensor data. When CAG is applied, the sink broadcast

a query to let all the nodes form into a group of clusters. Within each cluster, if the value of the cluster header is CR and the value for any node in the cluster is MR, then CR and MR satisfy $CR - CR \times \tau \leq MR < CR + CR \times \tau$, which means that the values within each cluster are similar. Subsequently, in the data collection phase of CAG, only cluster headers report their readings to the sink while the other nodes are suppressed.

For comparison purposes, we implement CAG in *ns-2* and test its performance by using the same three datasets. Since CAG is essentially an in-network sampling algorithm with similar functionality to our local suppression algorithm, our comparison is focused on how sensor nodes are selected to report their readings. In Figure 12 and 13, we show the sensor nodes that are selected to send their reports by the two algorithms respectively. In both figures, black dots represent the sensors chosen to send reports while the white ones are silent. By comparing Figure 12 and 13, we can see that when CAG is applied more nodes close to the sink are selected to send reports while nodes faraway from the sink are not sufficiently sampled. This phenomenon is caused by the drill-down mode adopted by CAG. More precisely, CAG let the sink flood a query to the entire network and expects this flooding can build a tree to reach every node. When packet loss ratio is high, the query may be dropped in the middle of delivery, thus those nodes faraway from the sink may not participate in the clustering process and none of them send reports. In contrast, our proposed local suppression algorithm is fully distributed and does not rely on communications with the centralized sink. As we can see, using our algorithm leads to a more evenly sampled sensor field (Figure 13) than using CAG (Figure 12). We can expect a better contour after performing the interpolation and smoothing algorithm as well.

IX. Conclusion

Wireless sensor networks hold great promises for monitoring the environments and providing timely samples of unusual events or the network itself. Moreover, diagnosis of faulty and energy-depleting sensors is critical to the health of the sensor network. Current solutions are unable to simultaneously achieve the goals of adaptive and timely sampling, robust monitoring, at low communication cost.

In this paper, we propose to use contour maps to effectively balance between these different goals. Our solution allows for progressive sampling of the field, and efficient local suppression of data. We devise algorithms to perform in-network data suppression along both spatial and temporal dimensions. We also describe algorithms on how to using interpolation and smoothing to reconstruct contours at the sink. The design can be applied in many scenarios, including spatial-temporal event monitoring, residual energy monitoring and faulty sensor detection. Simulation results confirm the effectiveness of our solution in terms of data reduction, accuracy and energy savings.

References

[1] http://www.cens.ucla.edu/seminars/slides/oct_17_han.ppt.

[2] http://www.cs.berkeley.edu/awoo/smartdust.

[3] S. Bandyopadhyay and E. Coyle. Spatio-temporal sampling rates and energy efficiency in wireless sensor networks. In *IEEE INFOCOM*, 2004.

[4] B. Bloom. Space/time trade-offs in hash coding with allowable errors. *CACM*, 13(7):422–426, 1970.

[5] C.Schurgers, V.Tsiatsis, S.Ganeriwal, and M.Srivastava. Topology management for sensor networks: Exploiting latency and density. In *ACM MOBIHOC*, 2002.

[6] S. Ganeriwal, C. C. Han, and M. Srivastava. Spatial average of a continuous physical process in sensor networks. In *Poster in ACM Sensys*, 2003.

[7] D. Ganesan, B. Greenstein, D. Perelyubskiy, D. Estrin, and J. Heidemann. An evaluation of multi-resolution storage for sensor networks. In *Proceedings of the first international conference on Embedded networked sensor systems*, pages 89–102. ACM Press, 2003.

[8] D. Ganesan, S. Ratnasamy, H. Wang, and D. Estrin. Coping with irregular spatio-temporal sampling in sensor networks. *SIGCOMM Comput. Commun. Rev.*, 34(1):125–130, 2004.

[9] A. Giridhar and P. R. Kumar. Data fusion over sensor networks: Computing and communicating functions of measurements. Submitted to Journal on Selected Areas in Communications, December 2003.

[10] T. He, C. Huang, B. M. Blum, J. A. Stankovic, and T. Abdelzaher. Range-free localization schemes for large scale sensor networks. In *Proceedings of the 9th annual international conference on Mobile computing and networking*, pages 81–95, 2003.

[11] J. M. Hellerstein, W. Hong, S. Madden, and K. Stanek. Beyond average: Toward sophisticated sensing with querie. In *Information Processing in Sensor Networks (IPSN)*, pages 63–79, 2003.

[12] C. T. Inc. http://www.xbow.com/.

[13] L.Doherty and K.S.J.Pister. Scattered data selection for dense sensor networks. In *Information Processing in Sensor Networks (IPSN)*, 2004.

[14] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. Tag: a tiny aggregation service for ad-hoc sensor networks. In *Proceedings of OSDI*, 2002.

[15] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. Tag: a tiny aggregation service for ad-hoc sensor networks. *SIGOPS Oper. Syst. Rev.*, 36(SI):131–146, 2002.

[16] ns2 network simulator. http://www.isi.edu/nsnam/ns/.

[17] S. Papadimitriou, H. Kitagawa, P. B. Gibbons, and C. Faloutsos. LOCI: Fast outlier detection using the local correlation integral. In *Proceedings of the 19th International Conference on Data Engineering (ICDE)*, pages 315–326, 2003.

[18] product specifi cation. http://www.rfm.com/products/data/tr1000.pdf.

[19] B. Przydatek, D. Song, and A. Perrig. Sia: secure information aggregation in sensor networks. In *Proceedings of the first international conference on Embedded networked sensor systems*, pages 255–265. ACM Press, 2003.

[20] rf monolithics. http://www.rfm.com.

[21] L. Vasudevan, A. Ortega, and U. Mitra. Application-specifi c compression for time delay estimation in sensor networks. In *Proceedings of the first international conference on Embedded networked sensor systems*, pages 243–254. ACM Press, 2003.

[22] Y.B.Ko and N.H.Vaidya. Location-aided routing (LAR) in mobile ad hoc networks. In *ACM MOBICOM*, 1998.

[23] W. Ye, J. Heidemann, and D. Estrin. An energy-effi cient mac protocol for wireless sensor networks. In *IEEE INFOCOM*, 2002.

[24] X. Yu, S. Mehrotra, N. Venkatasubramanian, and W. Yang. Approximate monitoring in wireless sensor networks. Submitted for publication, July 2003.

[25] J. Zhao, R. Govindan, and D. Estrin. Residual energy scans for monitoring wireless sensor networks. In *IEEE Wireless Communications and Networking Conference (WCNC)*, 2002.

[26] J. Zhao, R. Govindan, and D. Estrin. Computing aggregates for monitoring wireless sensor networks. In *First IEEE International Workshop on Sensor Network Protocols and Applications (SNPA)*, 2003.

[27] S.Yoon, and C.Shahabi. Exploiting Spatial Correlation Towards an Energy Effi cient Clustered AGgregation Technique (CAG). In *IEEE International Conference on Communications (ICC)*, Korea, May 2005.

[28] C. Guestrin, P. Bodik, R. Thibaux, M. Paskin, and S. Madden. Distributed Regression: an Effi cient Framework for Modeling Sensor Network Data In *Information Processing in Sensor Networks (IPSN)*, Berkeley, April 2004.

[29] S. Goel, and T. Imielinski. Prediction-based Monitoring in Sensor Networks: Taking Lessons from MPEG. In *ACM Computer Communication Review*, 31(5):82-98, October 2001.

[30] L. Rossi, B. Krishnamachari, and C.C.J. Kuo. Distributed Parameter Estimation for Monitoring Diffusion Phenomena Using Physical Models. In *First IEEE International Conference on Sensor and Ad hoc Communications and Networks (SECON)*, Santa Clara, CA, October 2004.

[31] S. Pattem, B. Krishnamachari, and R. Govindan. The Impact of Spatial Correlation on Routing with Compression in Wireless Sensor Networks. In *Information Processing in Sensor Networks (IPSN)*, Berkeley, April 2004.

[32] R. Cristescu, B. Beferull-Lozano and M. Vetterli. On Network Correlated Data Gathering with Explicit Communication: NP-Completeness and Algorithms. In *INFOCOM*, Hong Kong, 2004.

[33] A. Jindal, and K. Psounis. Modeling Spatially-correlated Sensor Network Data. In *First IEEE International Conference on Sensor and Ad hoc Communications and Networks (SECON)*, Santa Clara, CA, October 2004.

[34] Y. Yu, D. Ganesan, L. Girod, D. Estrin, and R. Govindan. Synthetic Data Generation to Support Irregular Sampling in Sensor Networks. In *Geo Sensor Networks*, Portland Maine, October 2003