Event Contour: An Efficient and Robust Mechanism for Tasks in Sensor Networks

Xiaoqiao Meng, Li Li, Thyaga Nandagopal, Songwu Lu
Technical Report TR-040018
Department of Computer Science
University of California, Los Angeles
Los Angeles, CA 90095
{xqmeng, slu}@cs.ucla.edu, erranlli@dnrc.bell-labs.com, thyaga@lucent.com

ABSTRACT

In large-scale sensor networks, due to the energy and communication constraints of each sensor, it is infeasible to collect event information from each individual sensor and process it at the sink. In this paper, we propose an efficient data-collection scheme that can be used for event monitoring and network-wide diagnosis. Our scheme relies on the well-known representation of data – contour maps, which trade off accuracy with the number of samples. To build the contour maps, we propose three novel algorithms: a distributed spatial and temporal data suppression algorithm, a reconstruction algorithm at the sink using interpolation and smoothing, and an efficient algorithm to convey routing information for extending data suppression over multiple hops. The error in data representation accuracy is bounded by a simple design parameter of our choice. By reducing the number of transmissions required to convey relevant information to the sink, the contour mapping strategy conserves power at sensor nodes and improves network lifetime. We also present novel and scalable security mechanisms to defend against the injection of forged reports. Our security mechanism uses location-based keys and is much more secure than existing schemes using node-based keys.

Our scheme is suited for accomplishing many tasks in sensors networks: (1) It presents a global picture of the network in both temporal and spatial domains. (2) It can be used as a diagnosis tool, e.g., to detect faulty sensors and to scan for residual energy. (3) It can work in concert with in-network aggregation schemes to further reduce the communication overhead of aggregation schemes. It requires only negligible processing and storage requirements in sensors, thereby 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 high packet loss rate in the network, and is robust to 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.

1. INTRODUCTION

Wireless sensor networks enable users to interact with the physical environment at an unprecedented level. When an event occurs in the field that a sensor network covers, e.g., temperature change, sensors will automatically report their observations on the event back to the data collection center (called sinks). Emerging applications include temperature and humility monitoring for fields of endangered plants, forest fire alarming in national parks, seismic monitoring and structure response, marine microorganism sampling, etc.

In this paper, we address the problem of monitoring and diagnosis in sensor networks. We seek to effectively monitor events covered by the sensor networks. In addition, we also examine the network health (e.g., the current energy level of sensors) and fault diagnosis (e.g., the damaged areas when part of the network is bombed by enemies), in order for the sensors to operate properly.

There are four main challenges for monitoring and diagnosis in sensor networks. First, an event may trigger widely varying readings at sensors spread over a large sensor field. In some cases, it is desirable to have a global view of the entire sensor field regarding that single event, and to monitor the change on a temporal basis. For example, when fire bursts out in a building, it might be necessary to continuously monitor all the sensors to observe the spread of the fire and track its intensity. Second, 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 fine-grained information with significant latency. For example, it is critical to know approximately how the fire is spreading and react quickly. Third, sensor readings tend to be noisy and some sensors even provide faulty readings. The solution to monitoring tasks must be robust against faulty or noisy sensors. Last, certain deployment environment can be hostile. Moreover, sensors may be subject to physical tampering. The solution must be robust to a small number of malicious sensors.

Previous solutions address only some of these factors. For example, Przyateck, Song and Perrig proposes secure aggregation solutions that compute simple functions on sensor readings, such as min., max, median. Their solution minimizes the amount of communication and is secure from tampering of a small number of sensors. However, the computed values are not applicable to many important sensing tasks, e.g. the fire event discussed earlier. The aggregation

is also vulnerable to the presence of noisy or faulty sensors.[1]

In order to develop a single solution that considers all the above factors and still perform tasks effectively, we introduce the notion of event contours. In principle, a contour is a line in a map that connects points of equal value. Neighboring lines 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 three fold: (1) we propose two novel algorithms: a distributed spatial and temporal data suppression algorithm, and a reconstruction algorithm at the sink using interpolation and smoothing. (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 filters and the knowledge of sensor locations at the sink to disambiguate potential routing paths (paths other than the actual is caused by the aggressive data reduction of Bloom filter). (3) We also present novel security mechanisms to defend against malicious attacks. Our security mechanism is based on location-based keys and is lightweight. Compared with previous solutions that only handle a fixed number of compromised nodes, our solution can potentially handle much larger number of subverted sensors.

The rest of the paper is organized as follows. We briefly describes the network model and assumptions in Section 2. We give an overview of our scheme in Section 3. We present the basic scheme in Section 4. We discuss important extensions in Section 5 and security in 6. We give the application scenarios in Section 7. We evaluate our scheme in Section 8. We conclude in Section 10.

2. NETWORK MODEL

We briefly describe the sensor network model and assumptions pertaining to our work.

We consider a network of *fixed sensor nodes* that are deployed in a 2 or 3 dimensional space. We define the space monitored by this sensor network as the *sensor field*. This sensor field is monitored by a set of monitor nodes, also known as *sinks*, which may lie either within or outside the sensor field[2].

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 also assume that the sink knows the topology of the sensor network, such as obstructions and physical geography of the sensor field. 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[3]. We do not require links to be bi-directional, however, we assume that the sink nodes are aware of any uni-directional links in the sensor field.

3. 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. However, it is not enough to report 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 1, 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 meet applications' need for an approximate view of the entire network with a desired granularity, we seek an efficient data representation that allows us to tradeoff accuracy and communication cost. More importantly, we would like to achieve this with minimal *computation* and *collaboration* requirements from sensors in the network. This allows us to avoid the overhead of coordination (e.g. synchronization, clustering) and achieve tasks with "dumb" sensors. Dumb sensors can be mass-produced cheaply which enables economical deployment of large-scale sensor networks.

We first elaborate the concept of *contours* that we propose to use for data representation. We then give an overview of the contour construction algorithm.

3.1 Data Representation using Contour Maps

Contour maps are a well-known technique for representing data when there exists a tradeoff between the desire to have more information and the cost of collecting 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

[1]We discuss more relate work in Section 9.

[2]Sink nodes that lie within a sensor field can be thought of as intermediate data collectors for a given region that can do some processing on the data before passing it to monitors outside the sensor field.

[3]This route must comprise of un-compromised nodes. See Section 6.

present a simple way of fine-tuning the tradeoff 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 temperature above a certain value.

3.2 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 without 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. Based on overheard information from neighbors, 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 its 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 tradeoff 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.

3.3 Security

A key problem in dense distributed sensor networks is to ensure that messages delivered to the sink are trust-worthy in the presence of malicious nodes. Malicious nodes can be legitimate sensors that are compromised. Malicious nodes can inject false reports into the system and can trick the sink into believing falsely that events are happening. We use a highly scalable, novel, location based key-binding algorithm, and propose an approach that allows sensor nodes to sign event reports. Only event reports that are signed by more than D sensors will be accepted by the sink (D is a configuration parameter). Our location-based scheme is much more secure against attacks using compromised nodes. Previous schemes [22, 27] allow attackers to forge event reports at any location as long as they can compromise D sensors and use the keys of these sensors to deploy malicious sensors at the given location. On the other hand, our scheme only allow them to forge event reports that go undetected at a given location.

4. ALGORITHMS FOR CONTOUR MAP CONSTRUCTION

For ease of description, in this section, we only describe the basic algorithm. We leave other optimizations to Section 5. In particular, we present the spatial suppression algorithm based on one-hop neighbors' reports, temporal suppression algorithm and the contour reconstruction algorithm at the sink using information sent by a subset of sensors in the network.

Let $M_e^u(t)$ denote the observed magnitude of an event e at sensor u at time t. We denote by $N(v)$, the network neighborhood of sensor v, the set of sensors that can transmit to v in a single hop. Let $\delta(\cdot)$ be the step-value function for the contour. $\delta(\cdot)$ can vary depending on the magnitude of an event. For ease of understanding, in this section, let δ be a constant.

4.1 Local Suppression at Each Sensor

A report in a sensor can be triggered by the occurrence of an event, 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.

4.1.1 Spatial Suppression

When a report generation is triggered at a sensor, it is possible that neighbors also have the same kind of trigger, especially if the trigger is from an event or a query. In order to avoid transient congestion in such situations and redundant transmissions, it is critical to let those sensors with more relevant information to transmit. Therefore, the sensors backoff for a period of time proportional to the relevance of their message.[4] Here, we assume that the relevance of a message is indicated by the magnitude of the event. Therefore, each sensor u backs off for a period of time inversely proportional to $1/M_e^u$.

If a sensor u that generates a report overhears some neighboring sensors in $N(v)$ transmit before it can, it computes the average of all the overheard readings that originated at its neighbors, M_e^{avg}. It compares M_e^{avg} and M_e^u. If the difference is less than δ, then

[4]This can be in addition to the MAC level back-offs.

u suppresses its report. The pseudo-code for this triggered report generation is shown below 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
```

Figure 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.

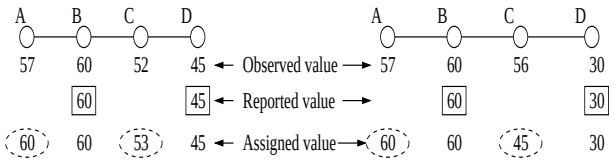

Figure 2: Spatial Suppression Example

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.[5]

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 to the maximum that 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.

[5]It is entirely possible that the route of B's report will be through C and D, forcing C to transmit and rendering C's suppression useless. However, the number of such nodes is usually $O(\sqrt{n})$, and therefore suppression will benefit other nodes not in the routing tree.

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 [12]. Routing path information can be conveyed to the sink, if needed, using path digests as discussed in Section 5.

4.1.2 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 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.

4.2 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.[6] 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.

4.2.1 Interpolation Algorithm

We assume the data sink knows the location of each sensor. This can be done through the localization process, e.g. [10]. We do not assume prior knowledge of the routing tree, since it is not necessary for the contour construction algorithm for interpolation based on one-hop neighbors' readings. We do assume the topology of the sensor network is known. 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 given event is defined

[6]Smoothing is done over sensors that are in the same physical region. This can be done easily at the sink, since it knows the geography of the region.

as I_{max}. From the definition, it can be seen easily that if $I_{max} = 0$, then there is no aggregation in the network, i.e., all sensors have reported their readings. If we assume that no reports are lost in transit, then $I_{max} \leq 1$.

Reports can get lost due to congestion and link errors along the path from sensor to the sink. Lost reports can result in holes in coverage, leading to a value of $I_{max} > 1$. To minimize interpolation errors due to losses, we need to constrain the maximum level of interpolation, I_p.

The event contour can now be constructed using the pseudo-code in Figure 3.

After receiving event reports at the sink,
1. Assign signal strengths M_e^u for black nodes
2. For each white node v, let status(v)=false
3. $I = 1$
4. while $I < I_p$
5. color(v) = black $\forall v$: status(v) $\neq$ false
6. for each white node v within I hops from a black node
7. let M_e^v be the average values of neighboring black nodes
8. status(v) = true
9. $I \leftarrow I + 1$

Figure 3: Contour reconstruction: interpolation at sink

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.

4.2.2 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. By using such a smoothing, an unknown reading of a sensor v is dependent on the readings of sensors that are more than one hop away from v. Hence, the reliability of the interpolation is improved. 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, n is the normalization factor, and σ controls the range of convolution. Suppose the reading value M^v of sensor v needs to be smoothed and L_r is the set of nodes r-hop away from node v. Then after smoothing,

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

The above distance-weighted sum only involves those sensors within 3-hop away from sensor v. This is 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.

5. EXTENSIONS

We discuss several important extensions in this section. We show how we enable multihop local suppression, how data aggregation is done with local suppression, how to suppress using information gain instead of step value.

5.1 Multihop Local Suppression

We have proposed local suppression algorithms that are based on the overhearing of neighbors. 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 are within h hops of itself (together with the report, a field indicating the number of hops of the report has traveled), and the report has a value that are 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.

5.1.1 Routing Information

To get more accurate and detailed contour information, we need to know the route from the data source to the data sink. If routing tree does not change, then we can attach this information in the packet once. However, routing paths (or tree) is maintained distributedly and can change as nodes die or go to sleep mode. Therefore, we need an efficient mechanism for the data sink to figure out the route. We propose a Bloom Filter technique. The technique works as follows.

The data source creates a Bloom Filter (a small bit array) [3] in the packet. Each node (including the data source) hashes their node ID into the bit array. We can select the number of hash functions to reduce false positive probability. A Bloom filter does not have false negatives. When the data sink receives the packet, it can apply the

same 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 value in the bit array (carried in the packet) are one, then the sink will declare that the node u is very likely to be on the route. Once the sink determines the potential candidate nodes set S, it will try to piece together the routes from the sink back to the source (reverse route) using the location of each candidate node (we assume the location of each sensor is fixed and the sink knows the locations of each sensor). If a node is far away, the sink will know that the node is a false positive. Therefore our mechanism is light weight and we can deal with false positives very well based on the location information. For example, if path length can be 20 hops, we have a 4-byte Bloom filter array and one hash function, then the false positive probability for one hop is $f = (1 - e^{-20/32}) = 0.46$. This is very large. However, a false positive path of 5 hops has a probability of $f^5 = 0.02$. 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.

5.1.2 Interpolation Algorithm at the Sink

When the sink receives the reports from the sensors, it iteratively perform 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 4. All we need to know is that whether they are direct neighbors of each other. For $i > 1$, we need to know how many hops two sensors are in routing paths (or tree). If two sensors are within i hops, then the one did not report is within δ of the sensor who directly reported its value.

5.2 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 easily accommodate function like max, min with the idempotent property (i.e. repeated apply the function to the same input, the function value is the same). To compute the max in network, if a sensor have a value lower than a value overheard, it just suppresses its communication.

To accommodate non-idempotent functions like average and sum, the aggregator needs to know the routing information. The routing information can be obtained efficiently as we described in previous subsection. An aggregator collects reports from sensors. It interpolates the values of sensors that are not received directly. The algorithm is 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.

5.3 Step Value vs. Information Gain

We have used step value of observed signal to suppress sensors from reporting too much information. Alternatively, we can use information gain instead. The idea is as follows. Suppose we would like to locate an event source. If a sensor's observed value reduces the uncertainty of the estimated event location by a threshold (or in other words, the information gain is greater than a threshold), then the sensor will incorporate its own reading into the estimation and send out the new estimated location.

The major issue are (1) how to represent information (or believe state, in general a probability distribution) since they need to be transmitted to other sensors; (2) how can we update the estimation incrementally with each sensor's observation. Chu et al. [4] gave an in-depth discussion on these issues. Nonparametric representation requires the transmission of each individual sensor's measurements as they are incorporated in the estimation of the belief state. To use nonparametric representation, sensors can estimate the information gain when it overhears the set of sensors' reading gets transmitted along the routing tree (suppose it has transmitted the reading). If the information gain exceeds a threshold, the sensor will send its reading to the sensor who is receiving the set of readings (towards the sink). Parametric representation can reduce redundant data transmission significantly. However, it relies on each sensor knows the parametric class of beliefs. In the case that the parametric belief is represented by Gaussian distribution, only the mean and covariance need to be transmitted. The Kalman filter equations are recursive update equations of the mean and covariance of the Gaussian distribution.

Multiple beliefs can reach the sink. The sink can merge these beliefs to have a more accurate estimations of the event source. In the case of ellipsoid representation of uncertainty, the intersection of the ellipsoids can be used as the target location.

6. SECURITY

The goal for security design is to ensure that reports about the physical world be generated faithfully, even in the presence of compromised nodes.

The threat model by compromised nodes is as follows. We assume that sensors are not compromised before their deployment. After being deployed, they are untethered and attackers can physically capture a node and compromise it to obtain all the information stored in it, including secret keys. However we assume that the time needed to physically capture and compromise a node is much longer than it takes sensors to finish certain bootstrapping tasks, including obtaining their locations and deriving security keys. Once a node gets compromised, attackers can use it to launch various attacks to the system. They can disrupt data forwarding by dropping, altering or replaying legitimate packets. They can also inject numerous bogus reports. Large numbers of injected reports can exhaust the energy and bandwidth of data forwarding nodes. Solutions to most attacks are relatively easy. Replay or message modification attacks can be addressed via standard security techniques. If a malicious sensor drops reports selectively, a sensor can randomly choose a trajectory [14] to the sink. The probability that not enough reports are received will be low. Therefore, the sink still gets enough information to reconstruct the contour with high probability. In this work, we focus on addressing the attack of injected false reports since it poses one of the most severe threats to event contour generation. If the sink can be tricked into believing some fake events are happening. This can result in very dire consequences. For simplicity, we only describe the mechanism that allows the sink to detect forged reports by up to D compromised sensors. To enable early detection of forged reports and drop them in-network. we need more complicated security mechanisms which will bring more overhead during normal operation. We omit them in this paper.

There are several goals for the security design. The solution must enable the sink to reliably detect and reject bogus reports even in the presence of large numbers of compromised nodes. It also has to be efficient to work with the resource constraints of sensor nodes, which preclude the use of asymmetric cryptography. Therefore, the design can only use inexpensive techniques such as symmetric key algorithms and hash functions. It should incur minimal computation, energy and storage overhead. Ideally, the overhead on individual nodes should be independent of the total number of node. This way, the design scales to a large sensor population.

In this paper, we explore a novel, location-based security approach that exploits the location-awareness of sensors to protect from the bogus reports fabricated by compromised nodes. In a typical sensor network, sensors use location information to tag all sensing report. Our security solution leverages this distinct feature of sensor networks to use location-based, rather than node-based, security primitives to protect the system. Fundamentally, the role of an individual sensor is minimized but the system functions are based on the collective behavior of multiple sensors in a given region. Whereas in conventional networks, messages are addressed to specific nodes, and security design is node oriented rather than location based.

Our location-based security solution is based on two fundamental premises. First, the security key owned by each node is a function of its location; each node has keys binded to its own, but not any other's location. Second, sensors are densely deployed so that each event occurring in the sensor field can be detected by multiple nodes, which makes it feasible to require that each event report be endorsed by credentials generated by multiple detecting nodes to minimize the possibility of bogus reports. An event without enough credentials will be rejected. Thus attackers cannot produce false reports about a specific location without compromising a sufficient number of nodes at that location.

The detailed design seeks to address several issues: (1) How does one generate location-dependent keys? The location alone cannot be used as keys, because attackers can easily forge such keys for any location. (2) How does one establish such keys in each node in an efficient and scalable way, given that the network may contain hundreds or even thousand of nodes? The ad hoc deployment precludes the possibility of knowing exact location of each node before deployment, while collecting all the nodes' locations to the sink after the deployment, then computing and sending keys to them does not scale to large networks. (3) How does one send credentials to the sink in an efficient and secure way? A naive way is to let each detecting node generate one credential and send one report separately. However, delivering such redundant reports from all detecting nodes would drain system energy excessively.

Generating Location-based keys We divide network field into virtual geographic square cells and bind the keys of each sensor to the cell it belongs to. The keys generated this way are called cell keys. We also assume that a system-wide master secret K_I. K_I is loaded into nodes before deployment and erased after bootstrapping; it is also known by the sink. Given the location of a geographic cell, expressed by coordinates $L_{i,j} = \{X_i, Y_j | X_i = X_0 + ic, Y_j = Y_0 + jc;\ i, j = 0, 1, 2, ...\}$, each node in the cell derives security keys as follows:

$$K_{X_i,Y_j,s} = H_{K_I}(X_i||Y_j||s),$$

where $H(\cdot)$ is a secure one-way function keyed with K_I, $||$ denotes concatenation, and s is a random number.

Credential announcement After agreeing on the report content, nodes should generate and announce their credentials to others, so that each can aggregate credentials independently. Each node uses its cell key of the cell (X_i, Y_j) to generate a MAC. Assuming it has $K_{X_i,Y_j,s}$,

$$MAC_s = H_{K_{X_i,Y_j,s}}(E||L||t).$$

where E is the event report, L is the location of the sensor and t is the time that the event is observed. The sensor should broadcast this MAC to all its neighbors. To avoid collisions, each node sets another random timer, upon the expiration of which it broadcasts a MAC announcement message:

$$\{E, L, t, s, MAC_s\}.$$

Others overhearing the message should record the tuple $\{s, MAC_s\}$. Since these MACs are used to prove that the event is real, we call them proof MACs. The event part $\{E, L, t\}$ is needed because more than one event might have been announced. Each node should clarify what report content it is endorsing.

Signing reports In this phase, nodes prepare and send the final report. Each node sets a random timer, upon the expiration of which it sends out the report with some MACs it gathers. The first node to fire the timer randomly picks D of the proof MACs it gathered and prepares a report message

$$\{E, L, t, MAC_{s_1}, MAC_{s_2}, ..., MAC_{s_D}, s_1, s_2, ..., s_D\},$$

where D is a design parameter, the number of credentials needed to make an event accepted by the user. It presents a tradeoff between overhead and security strength and should be set based on the deployment density. If a node hears a report message originating from a given number of hops away and its own event magnitude is within δ, then it will suppress its report.

Each overhearing node checks the D MACs against its list of gathered MACs. It keeps a counter for the number of MACs that are in its list and sent by others. Each time a report is sent, an overhearing node updates this counter. If this number reaches D, it cancels the timer. Because by then D MACs are sent and the event will be accepted by the user, there is no need to send redundant messages. Otherwise, its timer continues. Upon the timer's expiration, it prepares the report by picking D MACs from its list except those sent by others. Thus a legitimate node sends more MACs unless at least D MACs are sent.

Report verification at the sink When the sink receives the report, it can generate the cell keys used from L and $\{s_1, ..., s_D\}$. It then recomputes the MACs, compares the attached with them and counts how many are correct. The user makes the decision of whether to accept the event based on the number of correct MACs. Different approaches can be applied. One simple way is to set a threshold D. If this number reaches D, the event is accepted; otherwise it is rejected. Other ways such as setting confidence levels as functions of the number of correct MACs are also possible, but we do not elaborate on this.

For lasting events, the sink can accumulate the number of distinct correct MACs from continuous reports. The event is still accepted if the accumulated number satisfies the criteria.

7. APPLICATION SCENARIOS

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

7.1 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.

7.2 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.[7]

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 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.

7.3 Faulty Sensor Detection

Another aspect of sensor health monitoring is to determine the existence of sensors reporting faulty readings. The faults are due to problems with the sensors. Faulty sensors are defined as those sensor nodes which report data that is inconsistent with the observed event value at the sensor. Note that faulty sensors can be detected only when the value they send out is different from the observed value of the event.[8] We detect faulty sensors in two steps.

We first apply our contour mapping algorithm to the sensor field, as in the regular scenario. 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 sensors that report faulty readings that are significantly different from 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 to pinpoint any clusters with faulty sensors.

The second step uses well-known *outlier detection algorithms*, which help to pinpoint potential data which stand out from the rest of th data set. Various outlier detection algorithms exist, with the difference between the various algorithms being primarily in the implementation complexity. We use the outlier detection algorithm in [16] to detect the outliers in the contour data, and label them as faulty sensors.

8. PERFORMANCE EVALUATION

In our experiments, we use extensive scenarios to evaluate multiple aspects of the proposed contour mapping algorithms. These include the basic spatial and temporal suppression at individual nodes, the contour construction algorithm at the sink, and combining the basic design with outlier detection algorithms to identify faulty sensors, if any. The scenarios involve events that vary over both space and time. We also evaluate one important application of the proposed design, that is, monitoring residual energy in sensor networks.

8.1 Experimental settings

Our simulation platform is *ns-2* [15]. The basic physical layer parameters used in all our experiments are configured based on the *TR3100* transceiver [17], manufactured by RF Monolithics [19].[9] This transceiver has two operating modes, a high power mode and a low power mode. In the high power mode, the data rate is 576 kbps. Transmission and reception power requirements are $30mW$ and $21mW$ respectively. The high-power mode is used for data messages.

In the low-power mode, which we use for exchanging control messages, the data rate is 19.2 kbps. The power consumption in the low-power mode is 10% of the high-power mode operation. We also adopt the energy management scheme in [5]. In such a scheme, a node turns off its data-channel when it has no traffic, yet its control

[7] 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.

[8] Malicious sensors are discussed in Section 6.

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

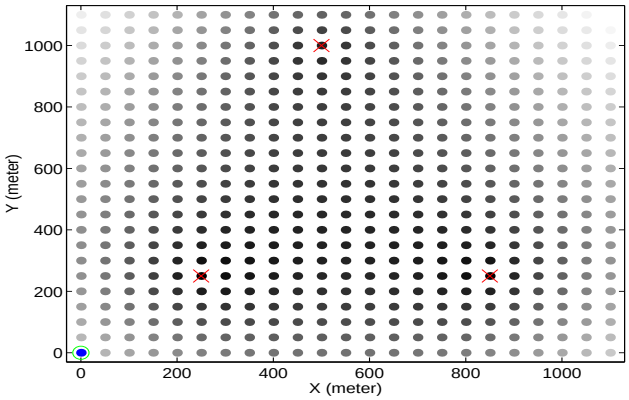

Figure 4: A 2D representation of actual event contour (Each dot represents a sensor node. $\times$ are the sources of diffusion model . The ◯ at the left-bottom corner is the sink. The value of gray color is scaled according to reading value, i.e., a deeper gray color represents a larger reading value.)

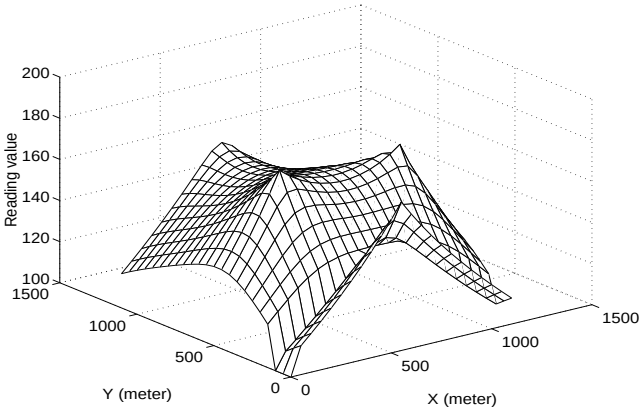

Figure 5: A 3D representation of actual event contour

radio is left on so the node can know when other nodes need it to forward traffic.[10] The transmission range for each sensor is 100 meters.

The MAC protocol used here is CSMA/CA, similar to IEEE 802.11. Control message exchange is performed via the control channel while the data packets are transmitted by the data channel. Since our design does not rely on the underlying routing protocol used, we assume that sensor nodes know the location of their sink and use a simple geographical routing protocol similar to LAR [21]. While this simple protocol does not produce the best delivery ratios possible, we use it as a baseline case. Using better routing protocols can only guarantee better performance with our algorithm.

We use diffusion models to represent the spatial and temporal spread of physical events [1] as perceived by sensor nodes. Diffusion models can be used to simulate the behavior of many physical processes, e.g., temperature diffusion from fire sources. In our study, a point p in the space has a value which is the summation of diffusion from multiple point sources, which is expressed by the following formula:

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

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 the point p and source i. a, k are distance factors. They are chosen to be 3 and 0.05 respectively. We simulate three point sources with their locations as $(250, 250)$,$(500, 1000)$,$(850, 250)$ (the unit is in meters). The values $M(i,t)$ of these sources are 145, 182, 160 units respectively.

In our study, we simulate a 23×23 grid topology. The distance between neighboring sensors is 50 meters. We assume there is only one sink. It is located at the left-bottom point of the topology. Given such a topology, the diffusion model and the sources, 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.

Unless explicitly mentioned, we assume $M(i,t)$ is time-invariant and hence the actual reading values for each sensor node is constant over time. In such a time-invariant scenario, we assume each sensor is only triggered by the sensing task once. We describe an example using spatial-temporal events later in the section. The suppression algorithm used in the examples in this section is the single-hop suppression algorithm described in Section 4. We consider the performance of the multi-hop suppression algorithm as part of future work.

For the given set of event sources, we compute the values observed by the sensors. The observed event magnitudes range between 102.7 and 182.3. We plot the sensor grid with the associated observed values in Figure 4. The figure shows the location of nodes, sink, and also the event sources. It uses grayscale depth to represent the reading value at each node. An alternative representation method for such a contour is 3D plotting. As shown in Figure 5, the reading values at sensor nodes are plotted by 3D meshes. In this paper, we mainly use the 3D plotting to depict event contours.

8.2 Performance of basic contour construction algorithm

Our first experiment demonstrates the accuracy versus power consumption tradeoff using our algorithm. We show that the contour mapping process is robust even when subject to heavy loss rates in the network.

We evaluate the accuracy of the contour maps constructed by the basic algorithms, assuming that no sensor reports are lost in the network. The spatial suppression algorithm is enabled and δ is set to be 20. The simulation result reveals that 92 nodes out of the 528 sensor nodes actually sent out their reading values. The location of these nodes are shown in Figure 6. The sink constructs the contour for the event based on these received readings. We plot the contour in Figure 7. We also plot the difference of the interpolated readings from the actual values in Figure 8. The figure shows that the maximum deviation in values as a result of the contour mapping process is 10, which is less than 7% of the average observed value.

[10] 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.

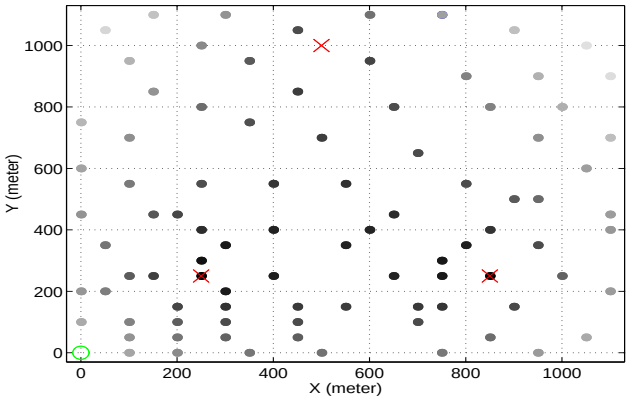

Figure 6: Nodes that actually sent out their reading values)

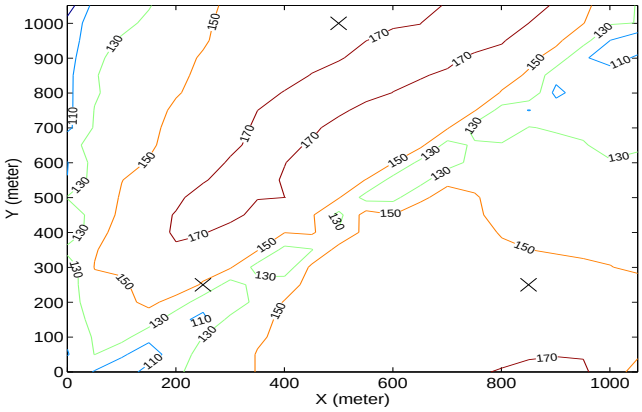

Figure 7: Two dimensional contour of sensor field

Note that the maximum error is the same as δ, the step-value for the contour.

The key observation is that the maximum deviation is related to the step-value chosen for the contour mapping process. This allows us to tradeoff the relative accuracy for the number of reporting nodes. In addition, the number of nodes that report is nearly an order of magnitude smaller than the total number of sensors, inspite of not using a clustering protocol. Using a clustering protocol will result in further reductions in the number of nodes that send out reports, without compromising the accuracy.

8.2.1 Fidelity of contour maps

In practice, reports can be lost on the way from the sensor to the sink due to either channel errors or congestion losses. In the above scenario, we introduce losses in the network by introducing congestion, which can be caused by heavy contention or repeated attempts to correct channel error. The average loss rate of reports is 20%. Out of the 92 reports sent by the sensors, 17 readings are dropped before they are received by the sink.

Based on these reports, the sink computes the contour. We plot the difference of the reconstructed contour from the original observed values in Figure 9. The figure shows that the maximum reading error brought by our protocols is 20, which is less than 15% of the average observed value at the sensors. We also compute the relative error for the recovered values at the sink and find that 98% sensor nodes have a relative error ratio less than 5%. The degradation in performance is marginal at best even at such high loss rates in the sensor network. This illustrates the robustness of our design.

8.2.2 Energy saving

We now evaluate the energy consumption of the proposed algorithm by comparing it to the scenario where there are no such mapping and all nodes transmit their reports. 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 report the results in Table 1. Energy needed to overhear neighbors forms the significant component of the overall energy consumed, as expected in wireless sensor networks. In each component, the

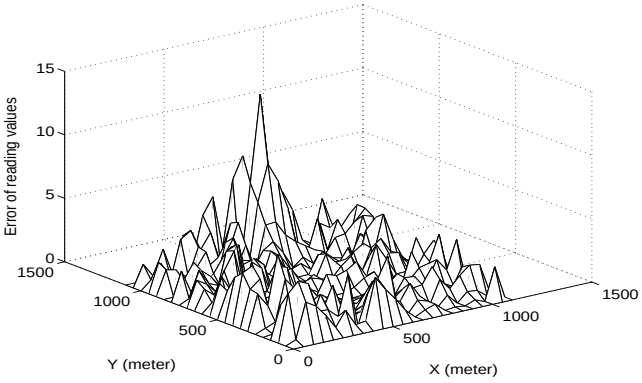

Figure 8: Error of recovered contour assuming no reports are lost

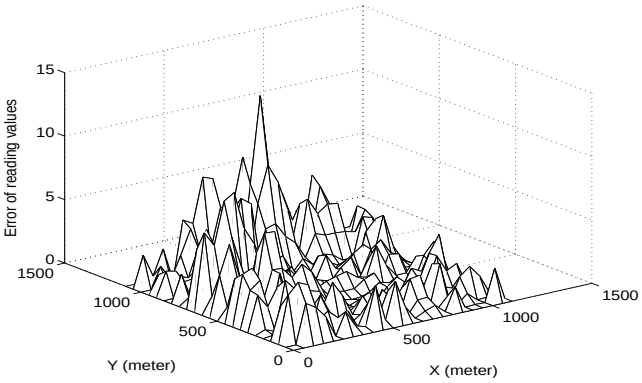

Figure 9: Error of recovered contour assuming 20% report loss

	Energy needed to (in Joules)		
	transmit	overhear	receive
No mapping	0.0018	0.0774	0.0011
Contour maps	2.1240e-004	0.0089	1.2716e-004
Energy saving	88.2%	88.5%	88.4%

Table 1: Energy savings using contour mapping - averaged over all sensor nodes

σ	1	3	5	8	11
Accuracy (in %)	97	95	89	80	65

Table 2: Accuracy of contour in the presence of Gaussian noise

power savings using the contour mapping scheme is upto an order of magnitude.

8.2.3 Sensitivity to Noise

In this example, we evaluate the sensitivity of the contour mapping algorithm to noise. We add Gaussian noise with zero mean and standard deviation, σ, to the sensor measurements in the entire network. We vary the noise variance and measure the accuracy of the mapping, defined as the *percentage of nodes having relative error* $\leq 5\%$, and tabulate the results in Table 2. The noise standard deviation, σ, is varied up to the $\delta/2$. It can be seen that the accuracy is pretty reasonably good for small values of σ relative to δ. When σ becomes comparable to δ the neighboring values vary more than delta, even though such noise levels are highly unlikely. This results in lot of transmissions, causing congestion in turn resulting in high levels of losses. Therefore the accuracy goes down.

However, if we define the accuracy threshold to be percentage of nodes having relative error $\leq 10\%$, then the accuracy is near 100% for the similar values of σ. This implies that the values assigned by the sink have an error of mostly between 5% and 10% compared to the original values. Thus, the contour mapping scheme is very robust in the presence of noise.

8.2.4 Sensitivity to δ

We study how sensitive the accuracy of the constructed contour is when use different δ. We still use the *percentage of nodes having relative error* $\leq 5\%$ to measure the accuracy and plot the result in Figure 10. From the figure, we see that the accuracy does not degrade too much when using various values for δ. Contrary to what one might expect, the accuracy stays constant after $\delta = 20$ because we apply suppression only over a single hop neighborhood. As a result, the number of reporting nodes remains fixed beyond a certain δ (here, beyond $\delta = 25$), since at least one node has to report in each one-hop neighborhood. When we use multi-hop suppression described in Section 5.1, the accuracy of the constructed contour will go down with step-size, since the number of nodes that send reports will also be severely limited.

8.3 Detection of faulty sensors

One of the applications of the contour mapping scheme is the ability to detect faulty sensors. We use a simplified version of the LOCI algorithm [16] to detect faulty sensors in the data set, as specified in Section 7.3. We randomly choose m sensor nodes and let each of them be assigned a random value which deviates more than 3δ from the actual event value at that point. Specifically, at the sink, we calculate the variance of each sensor's assigned value from that of its one-hop neighbors to detect the faulty nodes. Figure 11 plots the variance for a scenario when $m = 2$. The two peaks in the figure are located at $(250, 250)$, $(750, 400)$ and respectively correspond to the locations of the faulty nodes.

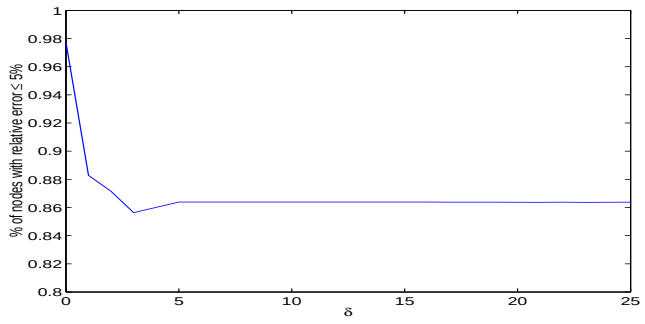

Figure 10: Accuracy of reconstructed contour vs. δ

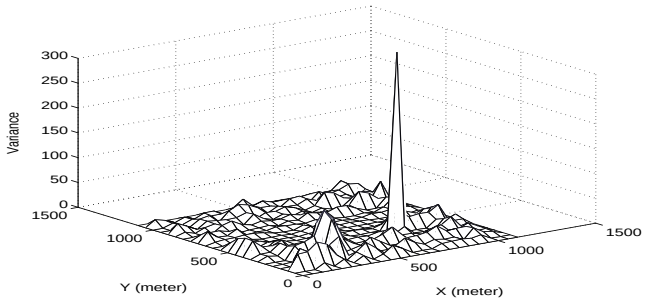

Figure 11: Variance of reading values at sensor nodes - the faulty sensors are at $(250, 250)$ and $(750, 400)$

8.4 Spatial-temporal event monitoring

To evaluate the performance of the combined spatial-temporal suppression algorithm, we simulate a time-varying event. 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 checks its 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 3 gives the number of reports actually sent out and received for each 200-second time interval. It can be seen that over each time epoch, the number of reports that are generated actually decreases. This decrease is due to temporal suppression. As a result, the delivery ratio of packets at the sink also improves tremendously from 80% in the first epoch to 91% in the fifth epoch. In all, 1637 reports are sent out. This 40% saving is due to both spatial and temporal suppression.

8.5 Contour maps for residual energy monitoring

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

Table 3: Results for temporal suppression (The first row gives the time epochs over which data is sensed at each sensor. The second row gives the total number of reports sent out during each time epoch. The third row gives the percentage of reports that are delivered to the sink.)

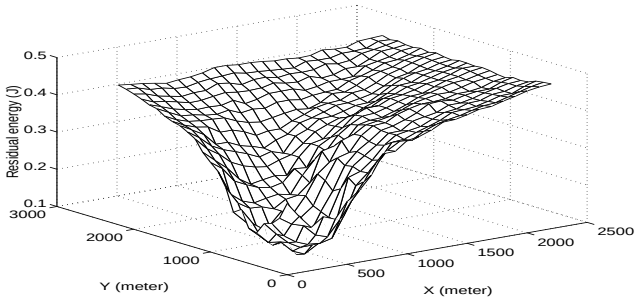

Figure 12: Contour of residual energy in sensor nodes

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 traffic.

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 7.2. 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 12. 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%.

9. RELATED WORK

We briefly review related work in this section.

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. Zhao et al. [26] propose a monitoring infrastructure that can identify system failures and resource depletion. They propose a novel 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. The sink knows the location of each sensor and the sink constructs the contour map.

Bandyopadhyay and Coyle addresses 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 of data-centric storage and routing, they propose the use of virtualization and boundary detection. Our techniques 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.

There are many other data reduction techniques in the literature. Some focuses on computing simple functions over the readings of sensor networks,e.g. [13, 18, 24]. Przydatek et al. [18] 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 [20]. These sophisticated techniques work well with large amount of data produced by sensors. They do not apply to sensor networks where the data is small. A storage and search system [7] is proposed as an efficient mechanism for sensor network applications. In-network wavelet-based summarization and progressive aging of summaries are used to support long term querying in storage and communication-constrained sensor networks. They show that these mechanisms support efficient drill-down search over summaries. Our techniques target sensor networks with different capabilities, i,e, sensors have very limited storage space and computational power. We argue that this enables economical deployment of large-scale sensor networks.

We believe the key security threat is that the sink can not distinguish correct reports from forged ones. The problem of detecting forged reports by up to D compromised sensors and dropping them has been studied in [22, 27]. If attackers can compromise D sensors, then they can deploy sensors at any given location (configured with keys recovered from those compromised sensors) to inject forged reports and can not be detected. Our location-based scheme is more secure. Attackers compromise D nodes can only forge reports on events in a certain location, not in any other locations.

10. CONCLUSION

Wireless sensor networks hold great promises for monitoring the environment and providing timely sampling of unusual events and the network itself. Moreover, diagnosis of faulty and energy-depleting sensors is critical to the health of the sensor network. However, current solutions cannot achieve the goals of adaptive and timely

sampling, robust monitoring, and 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 describe novel algorithms to perform in-network data suppression both spatially and temporally. In addition, the design allows for reconstructing contours at the sink using interpolation and smoothing, to protect the sensor values from malicious attacks. The design can be applied in many scenarios, and we use three examples — spatial-temporal event monitoring, residual energy monitoring and faulty sensor detection— to showcase the potential applications. Our simulation results confirm the effectiveness of our solution in terms of data reduction, accuracy, energy saving.

11. REFERENCES

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

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

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

[4] M. Chu, H. Haussecker, and F. Zhao. Scalable information-driven sensor querying and routing for ad hoc heterogeneous sensor networks. *Int'l J. High Performance Computing Applications*, 2002.

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

[13] 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.

[14] D. Niculescu and B. Nath. Trajectory based forwarding and its applications. In *ACM MOBICOM*, pages 260–272, 2003.

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

[16] 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.

[17] product specification. http://www.rfm.com/products/data/tr1000.pdf.

[18] 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.

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

[20] L. Vasudevan, A. Ortega, and U. Mitra. Application-specific 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.

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

[22] F. Ye, H. Luo, S. Lu, and L. Zhang. Statistical en-route detection and filtering of injected false data in sensor networks. In *IEEE INFOCOM*, 2004.

[23] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient 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. 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. Zhu, S. Setia, S. Jajodia, and P. Ning. An interleaved hop-by-hop authentication scheme for filtering false data in sensor networks. In *IEEE Symposium on Security and Privacy*, pages 260–272, 2004.