In-Network Detection of Anomaly Regions in Sensor Networks with Obstacles*

Conny Franke[1] Marcel Karnstedt[2]
Daniel Klan[2] Michael Gertz[3] Kai-Uwe Sattler[2] Wolfram Kattanek[4]
[1]Department of Computer Science, University of California, Davis, U.S.A.
[2]Databases and Information Systems Group, Ilmenau University of Technology, Germany
[3]Institute of Computer Science, University of Heidelberg, Germany
[4]Institute of Microelectronics- and Mechatronics Systems, Ilmenau, Germany

Abstract: In the past couple of years, sensor networks have evolved to a powerful infrastructure component for monitoring and tracking events and phenomena in many application domains. An important task in processing streams of sensor data is the detection of anomalies, e.g., outliers or bursts, and in particular the computation of the location and spatial extent of such anomalies in a sensor network. In this paper, we present an approach that facilitates the efficient computation of such anomaly regions from sensor readings. We propose an algorithm to derive spatial regions from individual anomalous sensor readings, with a particular focus on obstacles present in the sensor network. We improve this approach by proposing a distributed in-network processing technique where the region detection is performed at the sensor nodes. We demonstrate the advantages of this strategy over a centralized processing strategy by utilizing a cost model for real sensors and sensor networks.

1 Introduction

Driven by major advancements in sensor technology, several sensor networks have been and are being deployed in various application domains such as the monitoring of traffic, buildings, rivers, and the environment in general. Typical examples for environmental monitoring include precision agriculture (e.g., observing the humidity of the soil) and monitoring particles in urban areas to react to changes in air quality measures. An important objective in processing sensor data is the detection of anomalies that occur, e.g., in the form of outliers or bursts. This kind of data processing and analysis not only reduces the volume of data reaching end user applications but it also simplifies the further processing and interpretation of the sensor data.

By analyzing individual and aggregated sensor measurements, one can obtain useful information about the locations where anomalous events and phenomena occur. Such location information then can be visualized on a map and interpreted for individual sensors. In particular it can be used to derive *anomaly regions*. Such regions are composed of neighboring

*This work was in part supported by the National Science Foundation under Award No. ATM-0619139 and by the BMBF under grant 03WKBD2B.

sensors that show anomalous readings and are combined to describe a polygonal anomaly region. Compared to information about only individual (anomalous) sensors, providing users with such region information, including their spatial extent, has several advantages.

- It represents a natural way of event aggregation and correlation as needed in many monitoring applications, such as impact analysis.
- The location information associated with sensor data and regions allows for a direct processing of the results, e.g., for tracking regions.
- By approximating the regions in the unobserved space between sensors exhibiting normal and anomalous readings, one can determine region boundaries that more closely reflect the true boundaries of an event detected by a group of sensors. For this, one can also take propagation characteristics of detected events as well as natural and artificial obstacles occurring in the sensor network region into account.

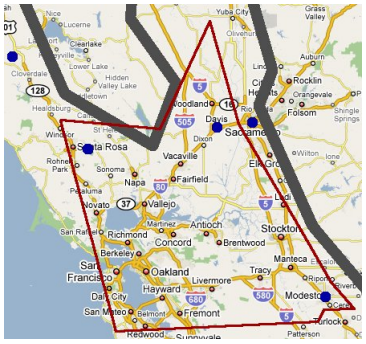


Figure 1: Example of anomaly region

For example, in the context of environmental monitoring this then not only allows for the detection of anomalies as measurement points on a map but also for determining regions where related anomalous values occur. Furthermore, obstacles such as buildings, ridges, rivers, and valleys are taken into account for predicting the propagation of anomaly regions. This aspect is illustrated in Fig. 1 where wind speed values are measured by the CIMIS sensor network [cim] in California. Based on the sensor locations (indicated by the blue dots) and the obstacles (here ridges indicated by thick gray lines) the marked region (read polygon) can be derived and placed on a map.

It should be noted, however, that the benefit of detecting anomaly regions and the underlying spatial aggregation of anomalous sensor readings can only be exploited if the event aggregation is performed locally at the affected sensor nodes or their close neighborhood, respectively. Particularly for wireless and battery-powered sensors, the expected reduction of expensive radio communication might improve the lifetime significantly when such a processing of sensor data is performed locally and intermediate results from groups of sensors are propagated in a hierarchical fashion. In this paper, we present such an approach for the distributed detection of anomalous spatial regions in sensor networks. In particular, the main contributions of our work are as follows:

1. We discuss a framework for anomaly detection that isolates the threshold-based region detection from the actual anomaly detection and, therefore, is orthogonal to the event detection that could be triggered by outliers or bursts.
2. In our approach for determining anomaly regions, we consider natural and man-made obstacles that might damp the effect of an event and thus need to be considered appropriately in determining the spread of (potential) anomaly regions.

3. We present a distribution strategy for the in-network detection and processing of anomalous sensor readings and deriving anomaly regions. This strategy can lead to significant savings in power consumption. We demonstrate the capabilities of the in-network detection approach using an evaluation based on real sensor network characteristics.

This paper is organized as follows: In Section 2, we introduce the scenario and goals of this paper. We also present our framework for the detection of anomaly regions. Section 3 summarizes related work in the areas of anomaly detection, region detection, obstacle handling, and in-network processing. In Section 4, we present our algorithm for detecting anomaly regions in the presence of obstacles. We discuss the benefits of the in-network computation of anomalies and anomaly regions in Section 5. The corresponding evaluation and experimental results are presented in Section 6. Section 7 concludes the paper.

2 Background and Setup

We assume a sensor network S comprised of m stationary sensors, $\mathcal{S} = \{s_1, \ldots, s_m\}$. Each sensor $s \in \mathcal{S}$ has a *spatial attribute*, $\langle x_s, y_s \rangle$, which defines its location in 2D space. Our approach is also applicable to a 3D setting, where nodes in the network are given by their x_s, y_s, and z_s coordinate to account for different elevations. For ease of presentation, we focus on 2D scenarios. The sensors are distributed non-uniformly in the network and monitor the *same* environmental *variable* such as temperature, humidity, or wind speed.

For a sensor s, a measurement of a variable is denoted $r_{s,t}$, with the timestamp t indicating when the variable reading was obtained. The network in our setting is *synchronized*, i.e., a set of m new measurements is processed in the network each time period. Synchronous processing is not a strict requirement for our method, but eases the processing of measurements as well as explaining the functionality of our technique.

Based on the spatial attribute of sensors a *spatial neighborhood* $N_f(s_i) \subseteq \mathcal{S}$ can be defined for each sensor $s_i \in \mathcal{S}$. A suitable neighborhood function f allows for different metrics, such as distance based neighbors (given a maximum distance r) or k-nearest neighbors.

2.1 Degree-Based Anomalies

Anomaly detection is a broad field that comprises areas like outlier detection, deviation detection, and burst detection. Anomalies of any kind are, by definition, data points that appear anomalous when compared to other data points in a data set or stream. For example, bursts are characterized as "abnormal aggregates in data streams" by Zhu et al. [ZS03]. An outlier is described as "a data point that is significantly different from the rest of the data points" in [BM07].

In *threshold-based* approaches, a threshold is used to separate two categories of data points, anomalous and normal ones. Some algorithms in the field of outlier detection use the notion of *degree-based outliers*, e.g., [FG08, WCD$^+$07], to better capture the intensity

of the observed anomaly. In this context, an *anomaly degree*, $AD \in [0, 1]$, is determined for each data point. By using an AD value to describe a data point, it is taken into account that some data points are more clearly anomalous than others. When analyzing a data stream, each sensor s and measurement $r_{s,t}$, respectively, is assigned a value $AD \in [0, 1]$, which can change with each new measurement the sensor obtains. An AD value of 0 indicates that the measurement obtained by s at time t is normal.

A *reference* is necessary to answer the question "$r_{s,t}$ is anomalous with respect to which other measurements?". In a spatial setting, it is common to use the spatial neighborhood $N_f(s)$ as reference. If only previous values of s are used to determine the AD of sensor s at time t, then $N_f(s) = \emptyset$. The other extreme is to set $N_f(s) = S$. Then measurements from all nodes in the network are used as reference. Between these two extremes, other definitions of $N_f(s)$ are possible, as mentioned above.

At time t, an anomaly detection algorithm is applied to each of the m new measurements. The output of the anomaly detection algorithm is a stream of tuples (s_i, t, AD), i.e., at time t sensor s_i has the anomaly degree AD.

In the following, we use two different approaches for anomaly detection, a degree-based outlier detection algorithm [FG08] and a burst detection algorithm [KKPS08]. Both algorithms determine the AD value of a measurement with the help of two threshold parameters k_{low} and k_{high}. If the measurement is between the two thresholds k_{low} and k_{high}, its AD value is computed based on its distance to k_{low}, i.e., the farther from k_{low} the measurement is, the higher is its assigned AD value. Otherwise the measurement is assigned $AD = 0$ or $AD = 1$ depending on whether it is above or below both thresholds.

2.2 Anomaly Regions and Obstacles

Anomaly regions are time-variant spatial regions in a sensor field where unusual phenomena or events are taking place at some point in time. Detecting event regions and their boundaries has been studied in, e.g., [FG08, KZ06], but so far obstacles in the sensor field have not been taken into account when constructing such regions.

For anomaly region detection we use the *TWISI* (Triangulated WIreframe Surface Intersection) approach proposed in [FG08], where polygonal anomaly regions are constructed with respect to an intensity threshold φ. At each point in time, the currently detected anomalous sensors are used for region construction. A user specified value $\varphi \in [0, 1]$ is used to select a subset of all detected anomalous sensors, i.e., only those sensors having $AD \geq \varphi$ should be included in an anomaly region. A region's boundary is placed in the unobserved space between anomalous and normal sensors. It is placed in such a way that we assume a measurement taken at a location next to the boundary would have an AD value close to φ. In Section 4 we briefly outline how region detection using the TWISI approach works.

We use the TWISI approach as the basis for our anomaly region detection because TWISI's boundary placement is very accurate. To illustrate this, we use the Intel lab sensor data [Int], which provide temperature measurements from 54 sensors deployed in the Intel Berkeley Research lab. Figure 2 shows a section from the region detected by TWISI. The black lines are part of the region boundary, and each of the sensors is labeled with its

current AD value. The gray sensor in the figure is a control point that does not contribute to the region boundary detection. It is used to check if the boundary placement is accurate. When setting the intensity threshold φ to 0.25, it can be seen that the control point having $AD = 0.27$ is located fairly close to the region boundary and inside the anomaly region. This shows that the boundary placement is meaningful with respect to the values that could be measured by new sensors, like the gray sensor in Figure 2, that are placed in the unobserved space between existing sensors, like the sensors having AD values 0.0 and 0.32 in Figure 2.

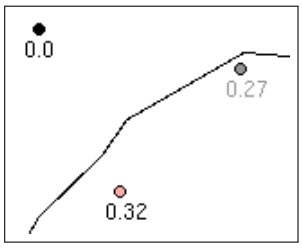


Figure 2: Accuracy of boundary placement

Obstacles in a sensor field are typically physical barriers like walls, buildings, rivers, or mountains. In 2D, obstacles are commonly modelled as simple polygons (see, e.g., [THH01]). Obstacles might damp the effects of a phenomenon, but do not necessarily stop its spread completely. A wall in a building will damp the effect of a cold room on the adjacent rooms, but the adjacent rooms' temperature will nevertheless be affected. In contrast, a draft in one room will not spread through walls to adjacent rooms. Thus, obstacles provide different damping factors for different phenomena. As the damping factor of large obstacles can vary, e.g., a mountain does not provide the same damping everywhere, we assign a damping factor $df(s_i, s_j) \in [0, 1]$ to each pair of sensor nodes, according to the obstacle(s) between the two sensors. Obstacles do not necessarily have to be physical barriers, as the air between two sensor locations can act as an obstacle as well, thereby damping the effect of an event due to the distance. Our approach is not limited to symmetric damping factors between two sensors, i.e., it is possible to define $df(s_i, s_j) \neq df(s_j, s_i)$.

By taking obstacles into account, we select a subset of all anomalous sensors detected at time t to be included in the anomaly region. This step uses the stream of anomalies as input, and works on a jumping window such that the most recent AD values of all m sensors are considered. The main purpose of detecting anomaly regions is to indicate the spread of events. We therefore use information about obstacles to extend the regions by also including anomalies having $AD < \varphi$. An anomalous sensor s with $AD_s < \varphi$ is included in a region if there is an obstacle between the source of an event and sensor s that damped the effect of this event. Assume $\varphi = 0.45$ and two sensors s_1 and s_2 with $AD_{s_1} = 0.29$ and $AD_{s_2} = 0.51$. Also assume an obstacle between s_1 and s_2 that incurs a damping factor of $df(s_1, s_2) = 0.2$. Sensor s_2 is clearly included in the anomaly region, as $AD_{s_2} \geq 0.4$. The event spreads from s_2 to s_1, but is damped by the obstacle. We therefore expect the AD value of s_2 to be lower than it would be without the obstacle, and decrease the threshold φ for including s_2 in the region by the damping factor. This step is called *threshold propagation*. By doing so, s_2 is now only required to have $AD \geq \varphi - df(s_1, s_2) = 0.45 - 0.2 = 0.25$ in order to be included in the anomaly region. As $AD_{s_1} = 0.29 \geq 0.25$, the detected region includes s_1 and s_2 This example is illustrated in Figure 4(b).

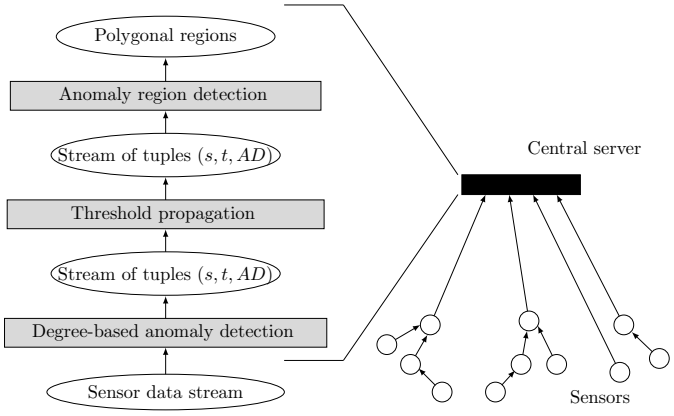


Figure 3: Conceptual and physical architecture of our framework

2.3 Three Tier Framework

All three steps, anomaly detection, threshold propagation, and anomaly region detection are combined into a three tier framework, as illustrated on the left hand side of Figure 3. Within this framework, the incoming stream of sensor measurements is piped through the different algorithms, which in the end output a stream of anomaly regions over time. Note that this framework comprises three modular processing steps, and therefore each of the three components can be replaced independently. For example, anomaly detection can be done using the burst detection or outlier detection approach mentioned above. Also, in an obstacle-free sensor field the second tier, threshold propagation, can be omitted without any changes to the remaining framework. The left hand side of Figure 3 shows the conceptual architecture of our framework, whereas the physical architecture is depicted on the right hand side. The latter consists of a hierarchically organized sensor network and a central server. Details about the physical architecture are presented in Section 5.

3 Related Work

Anomaly Detection Wu et al. [WCD$^+$07] propose a degree-based outlier detection algorithm for static data sets. Franke et al. [FG08] do the same for data streams. The output of such algorithms is the basis for the threshold propagation and region detection we propose in this paper. Other anomaly detection methods can be used as well, for example the burst detection algorithm proposed by Klan et al. in [KKPS08]. Their approach can be easily modified to detect degree-based anomalies by adding a second threshold k_{high} and computing AD values as described in Section 2.1. Similar modifications can be applied to other anomaly detection algorithms, e.g., [SPP$^+$06, ZS06].

Region Detection Other region or boundary detection algorithms in sensor fields, e.g., [DCXC05, KZ06], place the region boundary right next to the sensors that are on the edge of a region having distinct properties. Some papers, e.g., [DCXC05], define the region boundary as the set of sensors that are in the interior of a region but close to sensors outside the region. In contrast, our boundary placement is more considerate. We place the boundary between anomalous and normal sensors in a meaningful way, and its exact location depends in the intensity of an event at different locations.

Using spatial clustering algorithms, e.g., those mentioned in [HKT01], to partition the sensor field in anomalous and normal regions would not result in an accurate boundary placement either. This is because clustering aims at finding distinct groups of sensors rather than the exact location of the boundary between each two groups.

Obstacles Many publications deal with various data mining techniques in the presence of obstacles, e.g., [THH01, ZPMZ04]. However, in these methods obstacles are considered impenetrable objects that need to be bypassed, for example, to compute the distance between two objects as done in [ZPMZ04]. In contrast, we consider obstacles to be permeable albeit having different properties than their surroundings. We achieve this by defining a damping factor for pairs of sensors that are separated by one or more obstacles. This way, our definition subsumes existing definitions of obstacles, as a damping factor of 1 results in a impenetrable obstacle, providing absolute damping.

In-Network Computation TinyDB [GM04] and Cougar [YG02] are two well established query processing systems for sensor networks. Both systems support in-network processing with respect to data quality and sensor node life time. The essential difference is the used aggregation strategy. In TinyDB all sensor nodes are of the same type, whereas Cougar distinguishes three classes of nodes: sources nodes, intermediate nodes for data processing like aggregations, and gateway nodes, which connect the user. In order to decrease energy-consumption, both systems build aggregation trees to aggregate sensor data in nodes at higher levels within the routing tree. Building an optimal aggregation tree is NP-Hard. In [KEW02] the authors investigated the performance of aggregation in sensor networks and presented some heuristics to generate suboptimal aggregation trees.

Sensor placement in the network can have a significant impact on the communication costs of in-network processing. Dhillon et al. [DC03] propose an algorithm that places sensors in the network with the goal of effective coverage of the area. The sensor placement generated by the pSPIEL algorithm by Krause et al. [KGGK06] aims at minimizing communication cost between sensors and placing sensors at the most informative locations. In both papers obstacles are taken into account when finding the optimal sensor placement.

4 Detecting Anomaly Regions

The basis for our anomaly region detection is the TWISI method proposed in [FG08]. The TWISI approach assumes a barrier-free network, where events spread unhindered between

nodes. However, obstacles like buildings or mountains can obstruct the direct spread of temperature, wind, fine particles, etc. We therefore extend the TWISI approach to take obstacles into account. In the next paragraph, we describe the original TWISI method as proposed in [FG08], and then introduce our extensions.

The first step in TWISI is to construct a Delaunay triangulation of the sensor network using sensors as nodes in the triangulation. Then, a third dimension is added to represent the AD values of sensors, i.e., nodes are assigned a height according to their AD value. This results in a 3D surface, called triangulated wirefame surface or TWS for short, where outlier regions stand out as "hills". The height of each node is updated periodically when new measurements are obtained by the sensor and consequently its AD value is recomputed. To detect anomalous regions, a plane parallel to the x/y plane is intersected with the TWS at height φ, yielding a set of line segments where the plane intersects the different triangles of the triangulation. The projection of these line segments onto the x/y plane represents the boundaries of anomaly regions, which are polygons. The TWISI approach includes all anomalous sensors with $AD \geq \varphi$ in the generated regions.

Now, we show how to extend the TWISI approach to take obstacles into account. The goal is to propagate the original intensity threshold φ through the network such that also sensors having an $AD < \varphi$ might be included in the final anomaly region. This is motivated by the fact that the effect of an event might be damped by the obstacles in the network. By taking this damping factor betwen pairs of sensors into account, the anomaly region is extended such that we can observe the spread of a phenomenon taking the effect of obstacles into account. When propagating the threshold φ from s_1 to s_2, its value is lowered according to the damping factor between both nodes.

After the threshold is propagated through the entire network, the TWISI approach is applied. Due to the lowered threshold at some of the nodes, not one plane is used to intersect the TWS, but several planes at different heights, according to the threshold propagated to each of the sensor nodes. The resulting anomaly region is still a polygon, constructed from the line segments generated by the intersection of the planes at different heights with the TWS.

Propagation Algorithm The propagation algorithm works as described by Algorithm 1, and is iterative. All nodes that will be included in the anomaly region and their respective thresholds are stored in the data structure $\mathcal{O}$. In the initial iteration 0, we identify anomalies having $AD \geq \varphi$, add them to $\mathcal{O}$, and mark these sensors as visited by adding them to S_{marked}. We call these "level 0 anomalies", and their threshold is set to φ (lines 1 – 4). Then, in each subsequent iteration i the threshold is propagated from each node $o \in \mathcal{O}$ of the current level i to its direct neighbors, denoted $Neigh(o)$, i.e., all nodes that are connected to o by an edge in the triangulation of the network (line 7). This is done as long as new nodes are added to $\mathcal{O}$ in one iteration (line 5). If the neighbor n is an anomaly and has not been marked yet (line 8), the damping factor df between o and n is determined (line 10). The propagated threshold Δ of n is computed by subtracting the damping factor from o's threshold, i.e., $n.\Delta = o.\Delta - df$. If n is a direct neighbor of more than one level i anomaly, then we choose the largest of the propagated thresholds to prevent over-damping (lines 11 – 13). If n is not in $\mathcal{O}$ yet, i.e., it is not a direct neighbor of any of the level

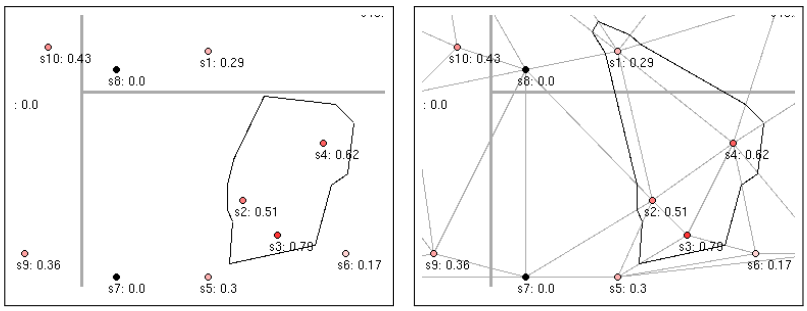
i anomalies checked so far, then n is added to $\mathcal{O}$ at level $i+1$ (lines 14 – 15). After checking all direct neighbors of all level i anomalies, we remove nodes o from $\mathcal{O}$ where the AD value is less than their propagated threshold Δ (line 16). This way, only nodes with AD value above the propagated threshold are included in the final anomaly regions.

```
   Input: φ
   Output: set of polygons
1  I = get-anomalies();                                   /* i ∈ I of the form [SID, AD] */
2  O = {[o.SID, o.AD, φ, 0] | o ∈ I ∧ o.AD ≥ φ};   /* o ∈ O of the form [SID, AD, Δ, lvl] */
3  S_marked = {o.SID | o ∈ I ∧ o.AD ≥ φ};
4  level = 0;
5  while ∃o ∈ O : o.lvl = level do
6      S_checked = ∅;
7      foreach o ∈ O : o.lvl = level do
8          foreach n ∈ Neigh(o) ∩ I : n.SID ∉ S_marked do
9              S_checked = S_checked ∪ {n.SID};
10             df = get-damping-factor(o.SID, n.SID);
11             if ∃o_n ∈ O : o_n.SID = n.SID then
12                 if o_n.Δ < o.Δ − df then
13                     o_n.Δ = o.Δ − df;
14             else
15                 O = O ∪ {[n.SID, n.AD, o.Δ − df, level + 1]};
16     O = O \ {o ∈ O | o.AD < o.Δ};
17     S_marked = S_marked ∪ S_checked;
18     level = level + 1;
19 return get-and-combine-line-segments(O);
```

Algorithm 1: Centralized threshold propagation and region detection algorithm

Marking visited sensors after each iteration prevents cycles, where the threshold of a node would initially be set in iteration i and then overwritten in iteration $j > i$ because of a chain of direct neighbors being included in $\mathcal{O}$. In combination with the iterative approach, marking visited sensors causes the threshold to be propagated to each node in only one iteration, and this iteration corresponds to the minimum number of hops from the level 0 anomalies. That is, each node is visited "as soon as possible", starting at the nodes that are initially above the threshold φ, and the propagated threshold for each node can not be overwritten in later iterations.

Figure 4 illustrates the effects of threshold propagation, using the example we already discussed in Section 2.2. The intensity threshold is set to $\varphi = 0.45$ in both figures. Each sensor is labeled with its sensor id and AD value. The triangulation of the nodes is shown in Figure 4(b) by the thin gray lines. The thick gray lines mark obstacles between sensors, which induce damping factors of 0.2 between each pair of sensors that is connected by an edge in the triangulation. Figure 4(a) depicts the anomaly region that was detected without threshold propagation. Sensor s_1 is not included in the region, although it is anomalous and fairly close to sensors that are inside the region, i.e., it is a direct neighbor of sensors s_2 and s_4, which are included in the anomaly region. Due to this proximity we would like to include s_1 in the region if its AD value, considering the damping factors to s_2 and s_4 respectively, is sufficiently high. This will be determined using threshold propagation.



(a) Without threshold propagation (b) With threshold propagation

Figure 4: Anomaly regions without and with threshold propagation

In Figure 4(b) threshold propagation was applied before constructing the anomaly region. The region in Figure 4(b) spreads to the area above the obstacle and includes the anomalous sensor s_1 there. This is what we wanted to achieve, as it provides us with additional information about the phenomenon we detected in the area below the obstacle. That is, the phenomenon spreads to sensors in the proximity of affected sensors in the lower area, i.e., to s_1, although s_1 is shielded from the phenomenon by an obstacle. In contrast, the region and thus the phenomenon does not spread to the area on the left of the obstacles, because the sensors s_7 and s_8 that are in the close proximity of the anomalies s_9 and s_{10} are normal. The phenomenon in the lower area cannot spread through normal sensors to the anomalous sensors. Technically speaking, sensors s_9 and s_{10} were not included in the region because they do not have a direct neighbor that has been added to the data structure $\mathcal{O}$ and thus could have propagated the threshold.

5 Distributed Approach

In wireless networks, sending and receiving messages is much more energy consuming than local processing. As energy consumption (measured in Joule J) is a crucial (if not the most crucial) cost factor in wireless networks, the number of messages should be minimized. In the centralized approach proposed up to here, all data sources, i.e., the sensors in the network, periodically send their data to a central server where it is analyzed and processed. Thus, a promising idea is to distribute the processing costs and by this hopefully lower the number of messages needed. This can be achieved by pushing (parts of) the processing steps down into the network, which is called *in-network processing*. Actually, there is a choice on the degree of distribution. As an opposite to the centralized processing, all processing steps are completely delegated to the sources and only detected anomalies are signalized to a central sever. We assume a multi-hop network having a hierarchical organization, similar to the one used in [SPP$^+$06]. The idea is to partition the network using virtual grids. The network has several levels: at the lowest level sensors in a local area are combined in one grid cell, and cells at higher levels subsume multiple cells from lower levels. At the highest level is one cell (the central server) representing the entire network. Each cell at each level (except the root cell) has a leader node which can be either cho-

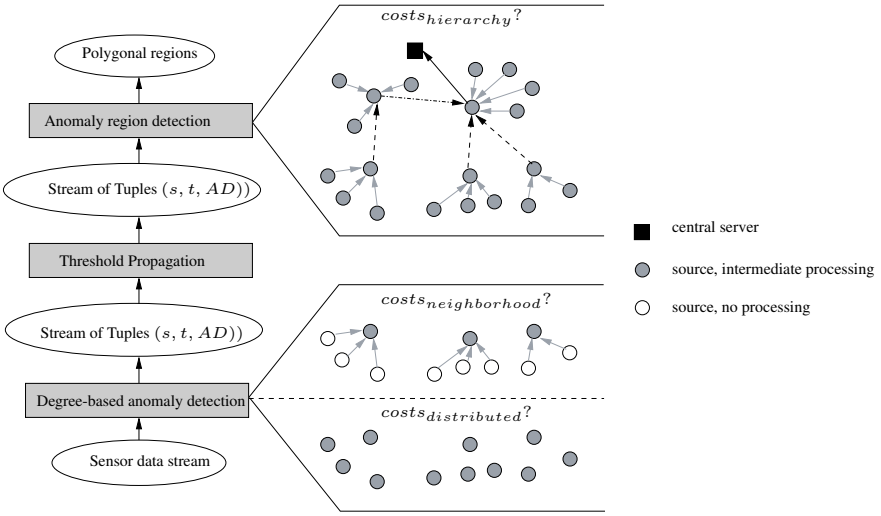


Figure 5: Possibilities of in-network processing

sen from the nodes in the network or it can be a virtual node. Like this, the hierarchy of nodes can be illustrated as a tree. In our setup we assume the same tree for both, multi-hop message passing and in-network processing (which is rather intuitive). This implies that each node can reach its parent node in one single hop. In [KGGK06] Krause et al. give an algorithm that can be used to partition the sensor network into grid cells and choose leader nodes. The resulting partition takes obstacles in the network into account, and thus it is unlikely that sensors within one cell are separated by one or more obstacles. Such an obstacle-aware partitioning of the network is desirable for our distributed algorithms.

Figure 5 illustratively summarizes the focus of the following section. For anomaly detection, we have three choices:

1. send all data to a central server for processing
2. choose leader nodes that collect data from all peers in their neighborhood and process the data
3. detect anomalies at each source separately

Option 3 is only practicable if anomalies are independent from neighboring sources, because otherwise a full exchange between all sources in a neighborhood is needed.

Threshold propagation and anomaly region detection cannot be processed on the individual sources, i.e., on the sensors, or for each neighborhood independently, because we also have to detect regions crossing neighborhoods. Thus, we only have the options:

1. send all data to a central server for processing
2. use a hierarchy between chosen leader nodes that exchange data accordingly

Obviously, threshold propagation and region detection can only be processed in-network if anomaly detection is done in-network as well. As all properties and statements made in the following equally apply to threshold propagation and region detection, from now on we use only region detection when referring to both methods, threshold propagation and region detection.

The choice on the degree of distribution depends on the trade-off between processing and transmission costs. For making the right decision on this, we will discuss an appropriate cost model. The crucial part is the energy consumption observed at the data sources and hierarchy peers. Thus, the factors influencing the total costs C (in $\frac{\mu J}{s} = W$ (Watt)) are:

- c_{msg}: constant costs for a single message (header etc.) in μJ
- c_{byte}: additional costs for each byte in a message in μJ
- $c_{cpu}(op)$: costs for processing operation op on a node in μJ
- r_m: the rate of taking measurements in $\frac{1}{s}$
- r_a: the rate of events, i.e., the average rate an anomaly is detected, in $\frac{1}{s}$
- m: number of sources contained in the network
- m_l: the number of leader nodes (the number of separated neighborhoods, respectively)
- h: average number of hops from a source to the central server (correlating with shortest paths in the node hierarchy)

In the following, we will develop general cost formulas for the different options of in-network processing. In Section 6 we will use concrete cost values in order to analytically evaluate the different choices.

5.1 Distributed Anomaly Detection

In the centralized approach, we consider the costs for transmitting data. The costs for processing at the central server are not the focus of this work, because we assume a powerful machine with external power supply for that. Usually, not every node is in radio range to the central server. Thus, messages are routed in a multi-hop manner using the hierarchy of nodes. Sending a message always results in a constant overhead c_{msg} due to header information etc. Additionally, costs depend on the size of the data contained, measured in bytes (c_{byte} for each byte). Receiving a message results in energy consumption as well. In our experiments, we observed that this is about the same costs as sending a message. See Section 6 for more details on this. A single measurement can be expressed using 2 bytes. Thus, we obtain the following costs for data transmission in the centralized approach:

$$C_{centr} = \underbrace{h \cdot r_m \cdot (c_{msg} + 2 \cdot c_{byte}) \cdot m}_{\text{send measurements}} + \underbrace{(h-1) \cdot r_m \cdot (c_{msg} + 2 \cdot c_{byte}) \cdot m}_{\text{receive measurements}}$$

Even if there exist techniques for collision prevention (based on time slots or ready/clear signals), there is a small probability of colliding messages. For convenience we omit this in our cost function, as it would only result in a small fraction of resent messages. Further, we assume all sources have the same periodicity, i.e., all sensors produce new measurements at the same frequency, and that messages are forwarded directly without collecting them at intermediate peers.

Several detection algorithms can be directly mapped to the data sources (of course, assuming that respective processing capabilities exist on the sensors). This holds, for instance,

for the introduced burst detection algorithm and the outlier detection if no information about the neighborhood is involved. For anomaly detection on the individual sensors, we obtain the following costs:

$$C_{anomaly} = \underbrace{r_m \cdot c_{cpu}(update) \cdot m}_{\text{update at sources}} + \underbrace{h \cdot r_a \cdot (c_{msg} + 2 \cdot c_{byte}) \cdot m}_{\text{send anomalies}} + \underbrace{(h-1) \cdot r_a \cdot (c_{msg} + 2 \cdot c_{byte}) \cdot m}_{\text{receive anomalies}}$$

Obviously, this can only help reducing energy consumption if r_a is significantly lower than r_m, which should be the usual case, as we are dealing with anomalies rather than normal situations. For popular sensors, c_{cpu} is orders of magnitude lower than c_{msg}.

If we take information about neighboring sources into account when determining anomalies, we make leader nodes responsible for detecting anomalies in each neighborhood. Sensors send messages containing single measurements to the leader nodes of their neighborhood and processing is done there. Then, we have

$$C_{lead} = \underbrace{r_m \cdot 2 \cdot (c_{msg} + 2 \cdot c_{byte}) \cdot (m - m_l)}_{\text{send and receive measurements}} + \underbrace{r_m \cdot c_{cpu}(update) \cdot m}_{\text{update at leader nodes}} + \underbrace{(h-1) \cdot r_a \cdot (c_{msg} \cdot m_l + 2 \cdot c_{byte} \cdot m)}_{\text{send anomalies}} + \underbrace{(h-2) \cdot r_a \cdot (c_{msg} \cdot m_l + 2 \cdot c_{byte} \cdot m)}_{\text{receive anomalies}}$$

As this assumes uniform distribution of all $r_a \cdot m$ anomalies over the m_l leader nodes, the costs C_{lead} represent an upper bound on the cost of anomaly detection at leader nodes.

The approaches for anomaly detection introduced in this work have no requirements regarding how much processing should be pushed into the in-network hierarchy. In fact, the processing can be totally distributed or is done at the individual leader nodes in case we have to handle neighborhoods. This does not hold for distributed region detection, where communication between leader nodes is mandatory. Depending on the structure and extent of a detected anomaly region, this can result in completely traversing a hierarchy of leader nodes potentially up to the central server on the very top of it. We discuss this approach of distribution and the corresponding costs in the following subsection.

5.2 Distributed Region Detection

At time t, the AD values of all sensors in one cell are collected at the cell's leader node. Then, threshold propagation is conducted as shown in Algorithm 2. At leader nodes of the lowest level, this algorithm works very similar to the centralized approach described in Algorithm 1, as can be seen in the comments below line 1 and line 22 as well as in lines 23–25 of Algorithm 2. Here, $\mathcal{I}$ contains only the outliers contained in the cell, not all outliers in the sensor network. All nodes that are direct neighbors of nodes in $\mathcal{O}$ but

Input: φ [, sets $\mathcal{O}, \mathcal{P}, S_{marked}, \mathcal{I}$ from sub-cells]
Output: set of line segments, $\perp$ if delegated to next level

1 **if** $\mathcal{P} = \emptyset$ **then** /* only possible at leader nodes lowest in hierarchy */
  /* fill $\mathcal{I}, \mathcal{O}$ and S_{marked} as in lines 1-3 of Algorithm 1 */
2 $\mathcal{P} = \emptyset$; /* $p \in \mathcal{P}$:= $[SID_1, SID_2, \Delta, lvl]$ */
3 **else**
4 merge sets $\mathcal{O}, \mathcal{P}, S_{marked}$ and $\mathcal{I}$ from all sub-cells;
5 $level = 0$;
6 **while** $\exists o \in \mathcal{O} \cup \mathcal{P} : o.lvl \geq level$ **do**
7 $S_{checked} = \emptyset$;
8 **foreach** $p \in \mathcal{P} : p.lvl = level$ **do**
    /* only possible at intermediate nodes at higher hierarchy levels */
9 **if** $p.SID_2 \notin S_{marked}$ **then**
10 **if** $\nexists s \in \mathcal{I} : s.SID = p.SID_2$ **then**
11 **if** $p.SID_2 \in LocalCell$ **then**
12 request $\mathcal{I}$ and S_{marked} from corresponding sub-cell and merge locally;
13 **else**
14 continue ; /* p is kept in $\mathcal{P} \rightarrow$ one level up */
15 df = get-damping-factor$(p.SID_1, p.SID_2)$;
16 **if** $\exists o_p \in \mathcal{O} : o_p.SID = p.SID_2$ **then**
17 **if** $o_p.\Delta < p.\Delta - df$ **then**
18 $o_p.\Delta = p.\Delta - df$;
19 **else**
20 $\mathcal{O} = \mathcal{O} \cup \{[p.SID_2, p.SID_2.AD, p.\Delta - df, level]\}$;
21 $S_{checked} = S_{checked} \cup \{p.SID_2\}$;
22 $\mathcal{P} = \mathcal{P} \setminus \{p\}$;
  /* expand current level as in lines 7-15 of Algorithm 1 */
  /* all nodes not in $LocalCell$ go into $\mathcal{P}$: $[o.SID, n.SID, o.\Delta, level + 1]$ */
23 $\mathcal{O} = \mathcal{O} \setminus \{o \in \mathcal{O} | o.AD < o.\Delta\}$;
24 $S_{marked} = S_{marked} \cup S_{checked}$;
25 $level = level + 1$;
26 **if** $P = \emptyset$ **then**
27 **return** *get-line-segments*$(\mathcal{O})$;
28 **else**
29 delegate $\varphi, \mathcal{O}, \mathcal{P}, S_{marked}, \mathcal{I}$ hierarchy upwards;
30 **return** $\perp$;

Algorithm 2: Distributed threshold propagation and region detection algorithm

are not in the local cell are collected in the set $\mathcal{P}$, which is later propagated upwards in the network hierarchy and the next higher leader node will attempt to determine the threshold for these nodes.

If $\mathcal{P}$ in the output of Algorithm 2 is empty, i.e., $\mathcal{P} = \emptyset$, propagation terminates and anomaly regions can be detected on this level of the network hierarchy. Otherwise, φ, $\mathcal{O}$, $\mathcal{P}$, S_{marked}, and $\mathcal{I}$ of the current leader node are sent upwards to the leader node of the next higher level. There, the incoming data sets from all sub-cells are merged (line 4). Then, all nodes that have been collected in $\mathcal{P}$ on lower levels are considered for insertion into $\mathcal{O}$. It is possible that the AD value of a node $p \in \mathcal{P}$ is not known to the leader node,

because either p is in a different cell on this level (line 14), or the sub-cell containing p did not send any data upwards. In the latter case, information about p is requested from the corresponding sub-cell (line 12). Generally, nodes in $\mathcal{P}$ are only considered for insertion into $\mathcal{O}$ if they have not been previously considered, i.e., if they are not in S_{marked}. In the distributed algorithm, this property results in a feature we call "neighborhood preserving". It means that if a node has been checked by a leader node on a lower level already, and is thus in S_{marked}, it will not be checked again at higher levels, even if this node appears in P with a lower level than in S_{marked}. This way, decisions made by sub-cells, i.e., the closer neighborhood of this node, about this node are not overwritten at higher levels. Insertion of nodes from $\mathcal{P}$ into $\mathcal{O}$ is similar to what happens in the centralized approach (lines 15–22 in Algorithm 2). As we stored the potential level for each node in $\mathcal{P}$, the nodes can be inserted at the appropriate level in $\mathcal{O}$. To propagate the threshold from nodes that have been newly inserted into $\mathcal{O}$ from $\mathcal{P}$, all nodes in the current level of $\mathcal{O}$ are checked again (comment below line 22).

For approximating the costs of in-network region detection, we have to introduce some more cost factors:

- L: average number of hierarchy levels involved in region detection
- m_{lR}: average number of nodes over all levels where (parts of) anomaly regions are handled
- m_{aR}: average number of anomalies handled over all levels
- m_{fR}: average number of anomaly regions detected and finalized over all levels – information about these regions is only forwarded following the multi-hop protocol
- $size_R$: average size of anomaly regions in bytes
- $size_{\{\mathcal{O},\mathcal{P},\mathcal{I}\}}$: average size of information needed to propagate regions upwards in the network hierarchy

This way, the number of regions is modeled by m_{aR}, m_{lR} and m_{fR}. The size of regions is modeled by m_{aR}, m_{lR} and L. Further, we assume that the anomaly regions are distributed uniformly over all cells. Based on these assumptions and the algorithm described above, we obtain:

$$
\begin{aligned}
C_{region} \quad = \quad & \underbrace{r_m \cdot c_{cpu}(anomalies) \cdot m}_{\text{anomalies at sources}} + \underbrace{r_a \cdot 2 \cdot (c_{msg} + 2 \cdot c_{byte}) \cdot (m - m_l)}_{\text{send and receive anomalies}} \\
& + \underbrace{r_a \cdot L \cdot (c_{cpu}(detect) \cdot m_{aR} + 2 \cdot (c_{msg} + size_{\{\mathcal{O},\mathcal{P},\mathcal{I}\}} \cdot c_{byte}) \cdot m_{lR})}_{\text{update and propagate regions}} \\
& + \underbrace{r_a \cdot (h \cdot (c_{msg} + size_R \cdot c_{byte}) + (h-1) \cdot (c_{msg} + size_R \cdot c_{byte})) \cdot m_{fR}}_{\text{forward finalized regions}}
\end{aligned}
$$

In this formula, we assume anomaly detection is done at the sources. If this is replaced by leader node-based detection, the r_a in the first line must simply be replaced by r_m (each measurement is sent, not only anomalies). $c_{cpu}(anomalies)$ corresponds to the CPU costs of the chosen method. Note that all listed cost formulas are worst case approximations, as

measurement	time	energy
compute average of 10 values	$52.3\mu s$	$0.272\mu J$
compute average of 100 values	$245\mu s$	$1.274\mu J$
single addition	$2\mu s$	$0.010\mu J$
single division	$27\mu s$	$0.140\mu J$
single multiplication	$16.2\mu s$	$0.08\mu J$
sending 1 byte	$4.85ms(2.33 - 6.95ms)$	$240.19\mu J(121 - 361\mu J)$
sending 10 bytes	$4.9ms(2.8 - 7.4ms)$	$252.93\mu J(146 - 385\mu J)$

Table 1: Average energy consumption measured on real sensors

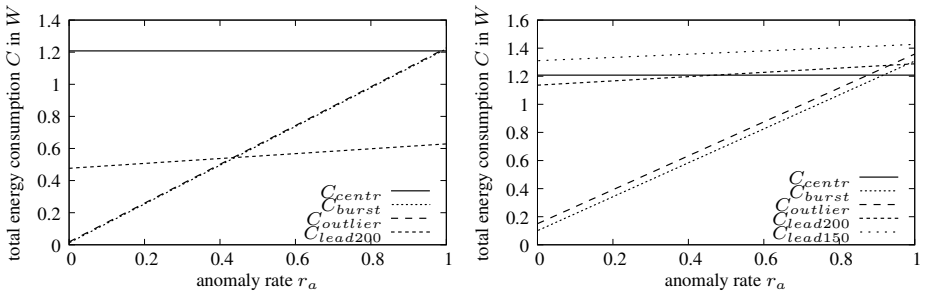
we use average values etc. Nevertheless, they are suited for analytically evaluating in-network processing by comparing the costs of each option. This is done in Section 6.

6 Evaluation

Due to space limitations, we focus the evaluation in this section on the distributed version of our approach. The centralized region detection approach has been evaluated in [FG08] and we showed the feasibility of the threshold propagation in Section 4. In the following, we present an analytical evaluation of the in-network processing options introduced in Section 5. For this, we instantiate the proposed cost formulas with values measured on real sensors and vary several cost factors. The purpose of this evaluation is *(i)* to identify the sensitive factors that have most influence on the actual choice, and *(ii)* to determine the benefits we get from in-network processing and in which situations. We expect the in-network methods being less energy consuming than the central approach up to a certain rate of anomalies r_a. The detection of anomalies on sources should perform best from this point of view, followed by the methods using leader nodes and hierarchy-based region detection.

We measured some typical Tmote Sky sensor nodes running TinyOS-1.x (16 bit microcontroller unit (MCU) MSP430F1611, 4 MHz clock rate, IEEE 802.15.4 compatible CC2420 transceiver with 250kBit/s). The MCU works on 16 bit integers, divisions are processed in software. For the sending operations of the transceiver we used maximal output power (+0 dBm). We assumed a battery voltage of 2.6 V and neglected any fluctuations that may occur in reality. All processing was done using the standard packet format of TinyOS-1.x, which means that for transmitting 1 byte raw data there are 12 bytes sent, due to headers, checksums etc. This corresponds to a raw sending time of about 0.384ms. Consequently, with 10 bytes raw data there are 21 bytes sent (about 0.672ms). Table 1 summarizes the most important results of the tests.

Roughly speaking, local processing is about 1000 times cheaper than communication. But, energy consumption of MCU operations is much more deterministic than communication in wireless networks. More complex routing protocols influence processing times and energy consumption. In our experiments, we did not apply such sophisticated protocols. Moreover, they would result in an overhead for both, processing and transmitting. The fluctuations observed for sending messages are due to the used CSMA protocol for radio transmissions, which uses random backup times among other things. Using TDMA this



(a) $w = 600$ (b) $w = 5000$

Figure 6: Varying anomaly rate r_a ($m = 1000, m_l = 200(150), h = 3, r_m = 1$)

would not apply – but in turn there would be more effort on synchronization etc. We also neglected situations of high load in the network, which could result in transmission delays as well. Further, we did not consider switching between active and idle modes and techniques for optimizing energy consumption in this case (e.g., by abstaining from the switch process in certain situations). Summarizing, we measured in a general but practically meaningful environment, which allows us to identify meaningful differences between the in-network options.

Based on these observations, we can instantiate the formulas from Section 5 with concrete values. For this, we derived the average values from Table 1. Interestingly, we observed that the number of sensors m and the rate of measurements r_m have no influence on the decision of in-network processing. Of course, they influence the total energy consumption, but all methods scale equally with them. The most influential factor clearly is r_a. This is illustrated in Figure 6. We show the costs in a $m = 1000$ sensor network, with a hierarchy depth of $h = 3$ and one measurement per second ($r_m = 1$). The costs of the central approach are C_{centr} and those of the anomaly detection on source level are C_{burst} and $C_{outlier}$ respectively. The anomaly detection method using leader nodes is referred to by C_{lead}. Figure 6(a) shows that message costs outweigh processing costs significantly. Only with highest anomaly rates the in-network costs are above the central costs. Neighborhood-based anomaly detection on leader nodes is cheaper for all rates. This is due to the implicit aggregation of sensor messages on the lowest level of the hierarchy.

The differences in Figure 6(b) show that all methods scale with the window size w, which is the size of the sliding window on which anomaly detection is done. Only the costs of the method on leader nodes are significantly affected by w. With $m_l = 200$ leader nodes the costs of the in-network method are lower than those of the central approach up to an anomaly rate $r_a = 0.5$ (which is still a very high rate). For larger neighborhoods ($m_l = 150$) and high values of w, the costs are higher even if there is no anomaly detected at all.

This indicates that another sensitive factor is the ratio of sensors to leader nodes, i.e., the size of the neighborhoods. To illustrate this, we vary this ratio in Figure 7. We used an anomaly rate $r_a = 0.1$. Further, we again show the effect of the window size w. The figure reveals that only for large window sizes (short terms up to an hour are common in streaming systems) the central approach should be preferred if neighborhoods are rather

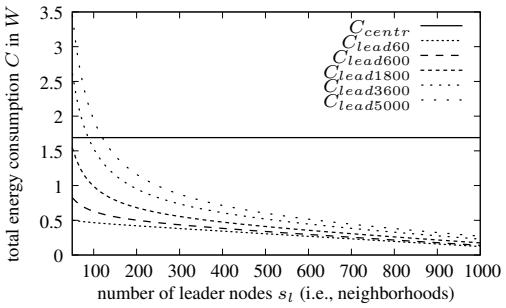
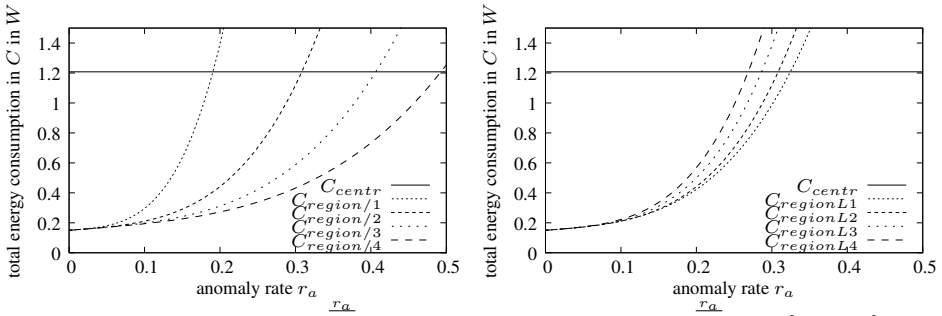


Figure 7: Varying number of leader nodes m_l ($m = 1000, h = 4, r_m = 1, r_a = 0.1$)



(a) $L = 2, m_{aR} = m \cdot \frac{r_a}{\frac{r_m}{\{1,2,3,4\}}}$ (b) $m_{aR} = m \cdot \frac{r_a}{\frac{r_m}{2}}, L = \{1, 2, 3, 4\}$

Figure 8: In-network region detection ($m = 1000, m_l = 200, h = 3, r_m = 1, w = 5000$)

large. The plots for different r_a with varying m_l look similar, but in contrast they are close for small m_l and differ more for large m_l – but not as significantly as for different w.

Figure 7 also shows the rather weak effect of increasing the depth of the hierarchy, i.e., average hop count. In contrast to Figure 6 we used a hop count $h = 4$. Clearly, the energy consumption of the central approach rises, caused by the multi-hop protocol. The in-network methods rise as well, but significantly slower. With 200 leader nodes, the energy consumption of the method on leader nodes is at about 1.14 for $r_a = 0.1$ in Figure 6(b). In Figure 7 the energy consumption for $w = 5000$ is at 1.17, whereas the central approach increased about 0.5 Watts.

Finally, we evaluated the in-network processing of anomaly region detection. As this is based on anomaly detection, we determined costs for both steps in conjunction (as we already did in the formula in Section 5). This only concerns the CPU costs for each method and influences performance of region detection negligibly. In our experiments we used the method for outlier detection on the sources exemplarily. It is rather difficult to identify suitable values for the used parameters m_{aR}, m_{lR} and m_{fR} without running tests on real data. However, the purpose of the cost comparison is to identify the sensitive parameters and to deduce the influence of region count and size. Thus, the effect of parameters is more important than their concrete values. Intuitively, all three depend on each other, and

all three depend on r_a as well. We tested a wide range of concrete relations and concluded that the most influencing parameters are m_{aR} and L. In Figure 8 we illustrate the effect of both. According to other tests we ran, a common average size of regions is about 6 sensors. Thus, we set $m_{lR} = m_{aR}/6$. The higher L, the larger are the regions and the smaller is m_{fR} for constant m_{lR}. We chose to use $m_{fR} = \frac{m_{lR}}{L}$.

Figure 8 shows that, as expected, energy consumption for in-network region detection is much higher than for pure anomaly detection. Furthermore, it does not scale linearly and the point of "break even" concerning the central approach is earlier. However, in-network processing is still worthwhile for rather small (and thus, usual) anomaly rates. The more regions occur (larger m_{aR}), the smaller these anomaly rates and the larger the increase of energy consumption (see Figure 8(a)). The size of the regions (larger regions result in higher values for L) has a significant influence as well, but not as much as the number of regions (see Figure 8(b)).

Summarizing, in-network processing provides an excellent opportunity to reduce energy consumption, and thus to increase life time of sensors. Anomaly detection on sources should be delegated in principle. If leader nodes are used to identify neighborhood-based anomalies, the choice should depend on the crucial parameters like the window size w. As expected, region detection can often be better performed at the central instance. But, for low anomaly rates it is still a good option for saving energy. This effect decreases with increasing number of regions and their size – due to the hierarchy-based approach.

7 Conclusion

Detecting regions of anomalous phenomena in sensor networks is an interesting and challenging task. In this paper we presented an anomaly region detection approach that is aware of the obstacles in the sensor field. The presented algorithm allows us to derive anomaly regions with meaningful boundaries instead of regions described only by grouping the measurement points. We use the notion of a damping factor between pairs of sensors to represent spatial obstacles like buildings or mountains. With the help of the damping factor we are able to describe the spread of a phenomenon though the sensor field taking the damping effect of obstacles into account.

Transmitting data within a sensor network is one of the most energy consuming sensor operations. In order to minimize communication costs and consequently improve the network life time, we also presented an in-network processing strategy for our detection approach. We developed a formal cost model for both the intuitive centralized approach and the complete in-network computing. Finally, we also showed analytical and experimental results to evaluate our approaches.

References

[BM07] Sabyasachi Basu and Martin Meckesheimer. Automatic outlier detection for time series: an application to sensor data. *Knowledge and Information Systems*, 11(2):137–

154, Feb 2007.

[cim] California Irrigation Management Information System (CIMIS). http://wwwcimis.water.ca.gov.

[DC03] Santpal Singh Dhillon and Krishnendu Chakrabarty. Sensor placement for effective coverage and surveillance in distributed sensor networks. In *Proc. of IEEE Wireless Communications and Networking Conference*, pages 1609–1614, 2003.

[DCXC05] Min Ding, Dechang Chen, Kai Xing, and Xiuzhen Cheng. Localized fault-tolerant event boundary detection in sensor networks. In *INFOCOM 2005: Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 2, pages 902–913, March 2005.

[FG08] Conny Franke and Michael Gertz. Detection and Exploration of Outlier Regions in Sensor Data Streams. In *SSTDM '08: International Workshop on Spatial and Spatiotemporal Data Mining in conjunction with ICDM 2008*, Dec 2008.

[GM04] Johannes Gehrke and Samuel Madden. Query Processing in Sensor Networks. *IEEE Pervasive Computing*, 3(1):46–55, 2004.

[HKT01] Jiawei Han, Micheline Kamber, and Anthony K. H. Tung. *Spatial Clustering Methods in Data Mining: A Survey*. Taylor and Francis, 2001.

[Int] MIT Computer Science and Artificial Intelligence Lab: Intel lab sensor data. http://db.csail.mit.edu/labdata/labdata.html.

[KEW02] Bhaskar Krishnamachari, Deborah Estrin, and Stephen B. Wicker. The Impact of Data Aggregation in Wireless Sensor Networks. In *ICDCSW '02*, pages 575–578, Washington, DC, USA, 2002. IEEE Computer Society.

[KGGK06] Andreas Krause, Carlos Guestrin, Anupam Gupta, and Jon Kleinberg. Near-optimal sensor placements: maximizing information while minimizing communication cost. In *IPSN '06: Proceedings of the fifth international conference on Information processing in sensor networks*, pages 2–10, New York, NY, USA, 2006. ACM.

[KKPS08] Daniel Klan, Marcel Karnstedt, Christian Pölitz, and Kai-Uwe Sattler. Towards Burst Detection for Non-Stationary Stream Data. In *KDML 2008: Knowledge Discovery, Data Mining, and Machine Learning*, Oct 2008.

[KZ06] Ivana Kolingerová and Borut Zalik. Reconstructing domain boundaries within a given set of points, using Delaunay triangulation. *Comput. Geosci.*, 32(9):1310–1319, 2006.

[SPP+06] Sharmila Subramaniam, Themis Palpanas, Dimitris Papadopoulos, Vana Kalogeraki, and Dimitrios Gunopulos. Online outlier detection in sensor data using non-parametric models. In *VLDB '06*, pages 187–198. VLDB Endowment, 2006.

[THH01] Anthony K. H. Tung, Jean Hou, and Jiawei Han. Spatial Clustering in the Presence of Obstacles. In *Proceedings of the 17th International Conference on Data Engineering*, pages 359–367, Washington, DC, USA, 2001. IEEE Computer Society.

[WCD+07] Weili Wu, Xiuzhen Cheng, Min Ding, Kai Xing, Fang Liu, and Ping Deng. Localized Outlying and Boundary Data Detection in Sensor Networks. *IEEE Transactions on Knowledge and Data Engineering*, 19(8):1145–1157, 2007.

[YG02] Yong Yao and Johannes Gehrke. The cougar approach to in-network query processing in sensor networks. *SIGMOD Rec.*, 31(3):9–18, 2002.

[ZPMZ04] Jun Zhang, Dimitris Papadias, Kyriakos Mouratidis, and Manli Zhu. Spatial Queries in the Presence of Obstacles. In *EDBT '04: 9th International Conference on Extending Database Technology*, pages 366–384, Heraklion, Crete, Greece, March 2004.

[ZS03] Yunyue Zhu and Dennis Shasha. Efficient elastic burst detection in data streams. In *KDD '03*, pages 336–345, New York, NY, USA, 2003.

[ZS06] Xin Zhang and Dennis Shasha. Better Burst Detection. In *ICDE '06*, pages 146–149, 2006.