CSRD (2009) 24: 153–170
DOI 10.1007/s00450-009-0063-y

SPECIAL ISSUE PAPER

In-network detection of anomaly regions in sensor networks with obstacles

Conny Franke · Marcel Karnstedt · Daniel Klan · Michael Gertz · Kai-Uwe Sattler · Elena Chervakova

Received: 10 October 2008 / Accepted: 11 March 2009 / Published online: 21 April 2009

Abstract In the past couple of years, sensor networks have evolved into an important infrastructure component for monitoring and tracking events and phenomena in several, often mission critical application domains. An important task in processing streams of data generated by these networks 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. Such information is then used as an important input to decision making processes.

In this paper, we present a novel approach that facilitates the efficient computation of such anomaly regions from individual sensor readings. We propose an algorithm to derive regions with a spatial extent from individual (anomalous) sensor readings, with a particular focus on obstacles present in the sensor network and the influence of such obstacles on anomaly regions. We then improve this approach by describing a distributed in-network processing technique where the region detection is performed at sensor nodes and thus leads to important energy savings. We demonstrate the advantages of this strategy over a traditional, centralized processing strategy by employing a cost model for real sensors and sensor networks.

Keywords Sensor networks · Data streams · Outlier detection · Distributed computation

C. Franke
Department of Computer Science,
University of California at Davis,
Davis, USA
e-mail: franke@cs.ucdavis.edu

M. Karnstedt · D. Klan · K.U. Sattler (✉)
Databases and Information Systems Group,
Ilmenau University of Technology,
Ilmenau, Germany
e-mail: kus@tu-ilmenau.de
e-mail: marcel.karnstedt@deri.org
e-mail: daniel.klan@tu-ilmenau.de

M. Gertz
Institute of Computer Science, University of Heidelberg,
Heidelberg, Germany
e-mail: michael.gertz@informatik.uni-heidelberg.de

E. Chervakova
Institute of Microelectronics- and Mechatronics Systems,
Ilmenau, Germany

1 Introduction

Driven by major advancements in sensor technology, 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. In applications like these, an important objective in processing sensor data is the detection of anomalies that occur, e.g., in the form of outliers or bursts of sensor readings. Focusing on such anomalies rather than all sensor data greatly reduces the volume of data reaching end user applications and it also simplifies the further processing of the sensor data. End user applications, in interaction with humans, are then used to make decisions based on the observed anomalies. Such decisions may range from near real-time notifications to the public, e.g., in case of hazardous events such as wildfires, flooding, or smoke plumes, to reconfigurations of the network by deploying additional sensors at locations close to where the anomalies have been observed.

By analyzing individual and aggregated sensor measurements, one can obtain important information about the locations where anomalous events and phenomena occur. Location information clearly is a key component in any decision making processes that employs sensor networks. Such location information can be visualized on a map and interpreted for individual sensors as well as groups of neighboring sensors. More importantly, and also the focus of our work, information about sensors that show anomalous readings can be used to determine *anomaly regions*. Such regions are composed of neighboring sensors that all show anomalous readings, and they are suitably aggregated to describe anomaly regions in the form of polygons. While there has been a significant amount of work on anomaly detection in sensor networks and data streams (see Sect. 3), only a few approaches also consider the derivation of region information from individual sensor readings. Compared to information about only individual (anomalous) sensors, providing users with such region information, including their spatial extent, clearly has several advantages:

- Anomaly regions represent a natural and intuitive way of event aggregation and correlation as needed in many monitoring applications, for example, in the context of impact analysis.
- The location information associated with sensor data and regions allows for a direct processing of the results, e.g., for tracking regions over time while new data from the sensor network are streaming in.
- By approximating the region boundaries in the unobserved space between sensors that show normal readings and those that show anomalous readings, one can determine 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. Obstacles obviously may have an impact on readings of neighboring sensors.

The following is an example that further illustrates the motivation of our work towards the computation of anomaly regions from individual sensor readings, here in the context of environmental monitoring. In such application scenarios, anomaly regions not only allow for the presentation of individual anomalous sensor readings as points on a map but they also describe where related anomalous sensor readings occur, here in the form of one or more spatial regions. This aspect is illustrated in Fig. 1. It shows part of the CIMIS weather station network in Northern California [2]. These stations record, among other values, wind speed and direction, temperature, humidity, and solar radiation.

The sensor locations are indicated by (blue) dots. What is shown too are *obstacles*, here ridges indicated by thick gray lines. Natural and artificial obstacles such as buildings, ridges, rivers, and valleys obviously need to be taken into account for determining region boundaries and the potential propagation of anomaly regions. In the figure, obstacles are indicated by thick (gray) lines. The resulting anomaly region, computed from the individual sensor readings and information about the obstacles, is represented by the thin (red) lines that make up a polygon. Obviously, the presentation of region information together with boundary sensors provides easier to interpret information than just individual (anomalous) sensors.

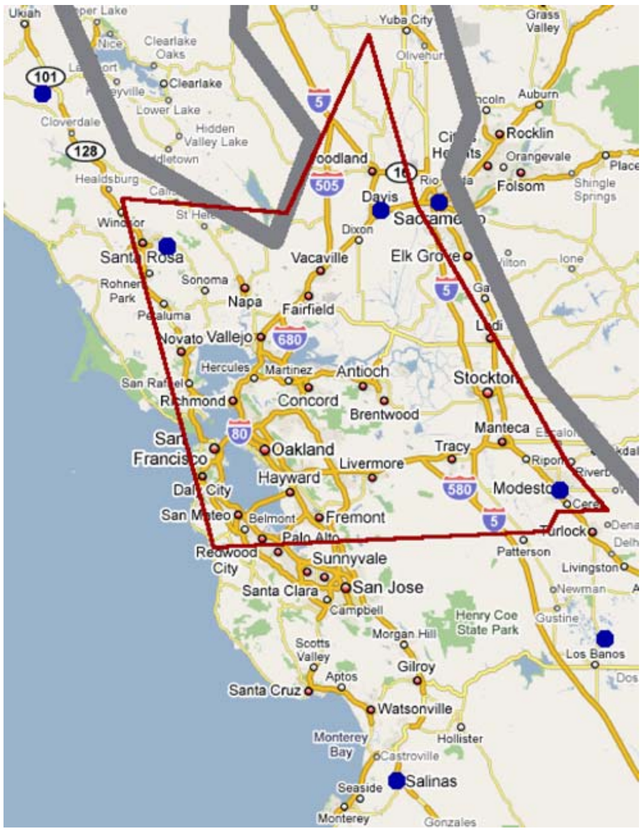


Fig. 1 Example showing sensors (*dots*), obstacles (*thick lines*) and resulting anomaly region (*polygon* described by *thin lines*)

For computing anomaly regions, obstacles clearly are an important aspect that needs to be taken into account when computing region boundaries or predicting the movement of an anomaly region over time. However, none of the existing approaches for detecting anomaly regions consider the aspect of obstacles. In our approach, we suitably model obstacles by means of *damping factors*, which describe the effect an obstacle has on neighboring sensors and which are input for determining region boundaries from anomalous sensor readings.

Another important aspect of practical relevance when processing and analyzing sensor data is the energy consumption of individual sensors and sensor nodes. This is especially true in the context of battery-powered, wireless sensor networks where data and messages need to be transmitted via radio communication among sensor nodes and applications. There are several works that describe energy saving approaches for processing data in sensor networks (see Sect. 3), which aim at extending the battery lifetime by sophisticated protocols. In such approaches, processing of sensor data is often performed locally and intermediate results from groups of sensors are propagated in a hierarchical fashion. As the detection of anomaly regions in sensor networks is a special case of data analysis, energy efficient approaches for determining such regions is of concern in our approach, too. In this paper, we therefore extend the basic region anomaly detection framework by introducing a distributed approach for region detection. A key aspect for our methodology here is that spatial aggregation of anomalous sensor readings and the detection of anomaly regions is much more efficient if the event aggregation is performed locally at the affected sensor nodes or their close neighborhood, respectively.

In summary, the main contributions of the work presented in this paper are as follows:

1. We present a framework for anomaly region detection in sensor networks that decouples the (typically threshold-based) detection of individual anomalous sensor readings from the anomaly region detection technique. Compared to existing approaches, this provides our approach with more flexibility in terms of employing alternative outlier or burst detection approaches for individual sensors.
2. In our approach for determining anomaly regions, we explicitly model and consider natural and man-made obstacles that might damp the effect of an event and thus need to be considered appropriately in determining region boundaries and the spread of (potential) anomaly regions.
3. We present a distribution strategy for the in-network detection and processing of anomalous sensor readings and the derivation of 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 remainder of the paper is organized as follows: In Sect. 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, the handling of obstacles, and in-network processing of sensor data. In Sect. 4, we present our techniques and 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 Sect. 5. The corresponding evaluation and experimental results are presented in Sect. 6. Section 7 summarizes the paper and outlines ongoing and future work.

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. Each sensor monitors environmental *variables* such as temperature, humidity, or wind speed. In the proposed framework, we assume one-dimensional sensor data, i.e., the *same* variable is monitored by all sensors. However, the proposed method can be extended to multi-dimensional data as well, e.g., by normalizing all attribute values with respect to their standard deviation and then computing a (weighted) sum of the values, as done by Anguilli and Fassetti in [3].

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. Depending on the type of sensors, such a period can range from a few seconds to several hours. Synchronous processing is not a strict requirement for our method, but streamlines the processing of measurements and eases the discussion of the functionality of our proposed techniques.

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. [30]. An outlier is described as “a data point that is significantly different from the rest of the data points” in [4].

Traditionally in anomaly detection algorithms, a binary decision is made about whether or not a data point is anomalous. This is called a *threshold-based* approach, because a threshold is used to separate two categories of data points, anomalous and normal ones. In contrast, some algorithms in the field of outlier detection use the notion of *degree-based outliers*, e.g., [10, 23], 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 obtained by the sensor. An AD value of 0 indicates that the measurement obtained by s at time t is normal. The AD value provides immediate feedback about the intensity of a phenomenon at a certain location. The change of a sensor's AD value over time indicates how the phenomenon evolves. By comparing the AD values of several sensors in a spatial neighborhood, detailed information about the distribution and spread of an event can be obtained.

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 indicated 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 anomaly degree AD. Thus, the anomaly detection algorithm can be seen as a membership function on the set $\mathcal{S}$ of all sensors, and it determines the membership of sensors in the fuzzy set of anomalous sensors at time t. Formally, the signature of the anomaly detection algorithm is defined as follows:

Input: stream of sensor measurements $r_{s,t}$, and for all sensors $s \in S$ their neighborhood $N_f(s)$
Output: stream of tuples (s, t, AD), where $AD > 0$.

In the following, we use two different approaches for anomaly detection, a degree-based outlier detection algorithm [10] and a burst detection algorithm [14]. 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 the thresholds.

Every anomaly detection algorithm matching the signature defined above can be used in our framework. In the following sections, we briefly introduce the two approaches we use for anomaly detection, one based on outlier detection and one based on burst detection.

Outlier detection. In [3] Angiulli and Fassetti propose a distance-based outlier detection algorithm for data streams, called STORM (STream OutlieR Miner). The algorithm works on a sliding window of the most recent sensor measurements. In distance-based outlier detection, the number k of measurements in the sliding window that have a distance of at most r from a sensor reading $r_{s,t}$ is determined, based on some distance measure. The STORM algorithm is threshold-based, i.e., it only distinguishes between normal and anomalous data points. Thus, if less than k measurements in the sliding window have a distance of at most r from $r_{s,t}$, then $r_{s,t}$ is anomalous, otherwise it is normal.

Franke and Gertz extended the STORM algorithm and propose a degree-based outlier detection algorithm for data streams in [10]. The technique is called *DSTORM*, standing for degree-based STORM. A sensor's AD value is determined based on a sliding window over the measurements of all sensors in $N_f(s)$. By adjusting the definition of a sensor's spatial neighborhood $N_f(s)$, DSTORM can use arbitrary references for computing a sensor's AD value.

The AD value of s at time t is then computed based on k. Lower values for k result in higher AD values, because the fewer measurements are similar to $r_{s,t}$, the more anomalous the measurement and thus the sensor itself is at time t.

Burst detection. Bursts are "abnormal aggregates in data streams" [30]. Detecting bursts is essential for identifying situations like a fire or heat-ups, where alarms should be triggered (for instance, to actuate a fire alarm or starting a sprinkler system).

The naive approach of inspecting all interesting window sizes in a data stream for a burst is not scalable and not suitable for data streams. Shasha et al. proposed in [20] an incremental one-pass algorithm, which can identify bursts over a variety of dynamically chosen window sizes. The presented technique is called *elastic burst detection*.

In [28] Zhang et al. introduced a synopsis structure, called *aggregation pyramid* and an algorithm for elastic burst detection. An aggregation pyramid is a triangular data structure over N stream elements with N levels. The presented aggregation pyramid offers some interesting properties. For instance, each subsequence of the data stream that passes through the bottom of the pyramid is covered by at least one node inside the pyramid. With this it is sufficient to check if the top of the pyramid succeeds a given threshold. If a burst is discovered a detailed search on the shadow of the cell that exceeded the threshold is necessary.

The original approach by Zhang et al. is only applicable to stationary data. The problem is that the threshold for identifying bursts is only adapted very slowly with evolving input data. This does not perform well in the presence of trends or periods. To dynamically identify a suitable threshold for triggering bursts, Klan et al. [14] used techniques known from time series forecasting (exponential smoothing) in order to compute a time series dependent threshold.

It is straightforward to extend the burst detection algorithm presented in [14] to detect degree-based bursts by computing two time series dependent thresholds instead of just one. The algorithm is designed to work on time series, i.e., streams of data from an individual sensor, and thus $\forall s \in S : N_f(s) = \emptyset$.

2.2 Anomaly regions and obstacles

2.2.1 Anomaly regions

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., [10, 15], 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 [10], 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 Sect. 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 [1], 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 at the right side of 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 Fig. 2, which are placed in the unobserved space between existing sensors, like the sensors having AD values 0.0 and 0.32 in Fig. 2.

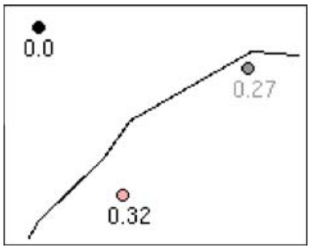


Fig. 2 Accuracy of boundary placement using $\varphi = 0.25$

2.2.2 Obstacles

Obstacles in a sensor field are typically physical barriers like walls, buildings, rivers, or mountains. In 2D, obstacles are commonly modeled as simple polygons (see, e.g. [22]), but can be as simple as line segments when modeling walls within a building. To accommodate a 3D setting, obstacles can be modeled as planes in 3D or complex polyhedra. In this paper, we model obstacles as line segments in 2D, although the use of polygons is possible as well. Polygonal obstacles would slightly increase the runtime of our algorithms due to the more complex computation when checking for intersections of outlier regions with obstacles.

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. This way, we are able to take geometric properties of the obstacles into account. 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 $AD_{s_2} \geq 0.45$. 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 Fig. 4b.

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 sliding window such that the most recent AD values of all m sensors are considered. The interface of this step is as follows:

Input: stream of tuples (s, t, AD), threshold φ, damping factors $\forall s_i, s_j \in S : df(s_i, s_j)$ and $df(s_j, s_i)$
Output: stream of tuples (s, t, AD) that is a subset of the tuples in the input stream.

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 Fig. 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 Fig. 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 Sect. 5.

We propose data stream processing on the basis of complex query plans. Each (mining) task is represented by an according operator. Thus, faulty sensor readings should be filtered by an operator for data cleaning preceding the operator for anomaly region detection.

3 Related work

Anomaly detection. Wu et al. [23] and Zhang et al. [26] propose degree-based outlier detection algorithms for static data sets. Franke and Gertz [10] 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 [14]. 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 Sect. 2.1.

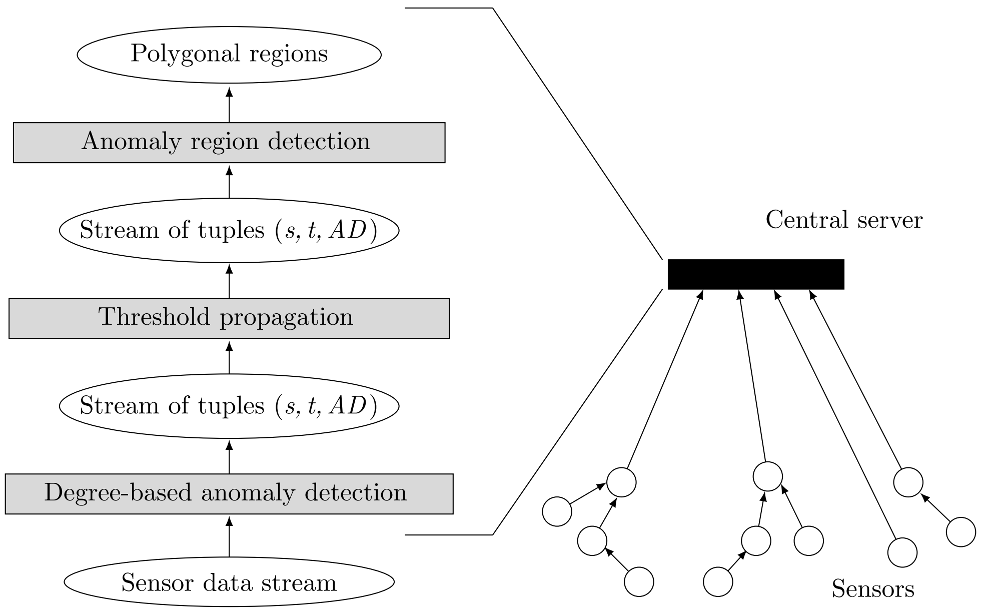


Fig. 3 Conceptual and physical architecture of our framework

Similar modifications can be applied to other anomaly detection algorithms, e.g. [21, 28].

Region detection. Existing work on region and boundary detection in sensor fields, e.g. [5, 8, 15, 18, 19] is not mainly concerned with the precise placement of the region boundary according to the intensity of an observed phenomenon. For example, the algorithms proposed in [5, 8, 15] place the region boundary right next to the sensors that are on the edge of a region having distinct properties. Some papers, e.g., [8], 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 discussed by Han et al. in [13], 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 pair of groups.

Obstacles. Many publications deal with various data mining techniques in the presence of obstacles, e.g. [9, 22, 27]. 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 by Zhang et al. in [27]. 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. Sensor networks presents a comparative new research area. In last years most of the researchers are focused on low level development, like the wireless communication and efficient routing protocols. The increasing spread of real deployments results in more focus of efficient data accumulation and computation. Several publications [6, 29] have shown that sending messages is one of the most power expensive operations. Main solutions to minimize expensive message sending is to compute data as soon as possible within the network.

TinyDB [11] and Cougar [24, 25] 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 aggregation strategy that is employed by these approaches. 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. Krishnamachari et al. [17] investigated the performance of aggregation in sensor networks and presented some heuristics to generate suboptimal aggregation trees.

Sensor placement in a network can have a significant impact on the communication costs of in-network processing. Dhillon et al. [7] 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. [16] 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 [10]. 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 [10], 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 the triangulated network 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 triangulation of the 2D sensor network is only computed once, as we assume the network to be somewhat stable. If nodes in the sensor field are added or removed frequently, the triangulation has to be recomputed on a regular basis to ensure that it reflects the spatial relations in the sensor network accurately.

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 between pairs of sensors into account, the anomaly region is extended such that one 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 below, and it 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 an AD value above the propagated threshold are included in the final anomaly regions.

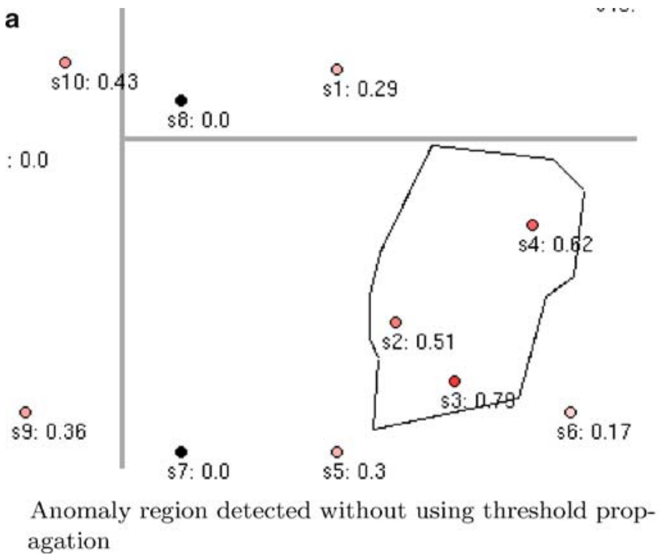
Marking visited sensors after each iteration prevents cycles, where the threshold of a node would initially be set in iteration i and then overwritten in iteration $j > i$ because of a chain of direct neighbors being included in $\mathcal{O}$. In combination with the iterative approach, marking visited 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.

```
   Input: $\varphi$
   Output: set of polygons
1  $\mathcal{I}$ = get-anomalies();   /* $i \in \mathcal{I}$ of the form $[SID, AD]$ */
2  $\mathcal{O} = \{[o.SID, o.AD, \varphi, 0] | o \in \mathcal{I} \wedge o.AD \geq \varphi\}$; /* $o \in \mathcal{O}$ of the form $[SID, AD, \Delta, lvl]$ */
3  $S_{marked} = \{o.SID | o \in \mathcal{I} \wedge o.AD \geq \varphi\}$;
4  $level = 0$;
5  while $\exists o \in \mathcal{O} : o.lvl = level$ do
6      $S_{checked} = \emptyset$;
7      foreach $o \in \mathcal{O} : o.lvl = level$ do
8          foreach $n \in Neigh(o) \cap \mathcal{I} : n.SID \notin S_{marked}$ do
9              $S_{checked} = S_{checked} \cup \{n.SID\}$;
10             $df$ = get-damping-factor($o.SID, n.SID$);
11             if $\exists o_n \in \mathcal{O} : o_n.SID = n.SID$ then
12                 if $o_n.\Delta < o.\Delta - df$ then
13                     $o_n.\Delta = o.\Delta - df$;
14             else
15                 $\mathcal{O} = \mathcal{O} \cup \{[n.SID, n.AD, o.\Delta - df, level + 1]\}$;
16     $\mathcal{O} = \mathcal{O} \setminus \{o \in \mathcal{O} | o.AD < o.\Delta\}$;
17     $S_{marked} = S_{marked} \cup S_{checked}$;
18     $level = level + 1$;
19 return get-and-combine-line-segments($\mathcal{O}$);
```

Algorithm 1 Centralized threshold propagation and region detection algorithm

Figure 4 illustrates the effects of threshold propagation, using the example we already discussed in Sect. 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 Fig. 4b 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 4a 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.

In Fig. 4b threshold propagation was applied before constructing the anomaly region. The region in Fig. 4b 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



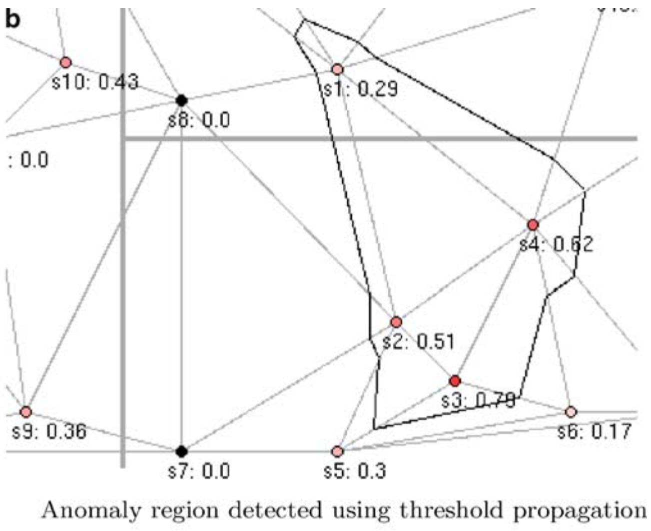


Fig. 4 Anomaly regions detected with and without using threshold propagation

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 [6]. 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 described above, 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 cost and by this hopefully lower the number of messages needed. This can be achieved by pushing (parts of) the processing steps 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 sensors and only detected anomalies are signalized to a central sever. As we will show, this is straightforward for anomaly detection. Other algorithms, like outlier detection comprising information from a neighborhood or region detection, can only be delegated to a set of (hierarchically organized) *intermediate peers*.

We assume a multi-hop network having a hierarchical organization, similar to the one used by Subramaniam et al. in [21]. 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 highest level) has a leader node, which can be either chosen 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 [16] Krause et al. give an algorithm that can be used to partition a given sensor network into grid cells and to 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 sensor separately.

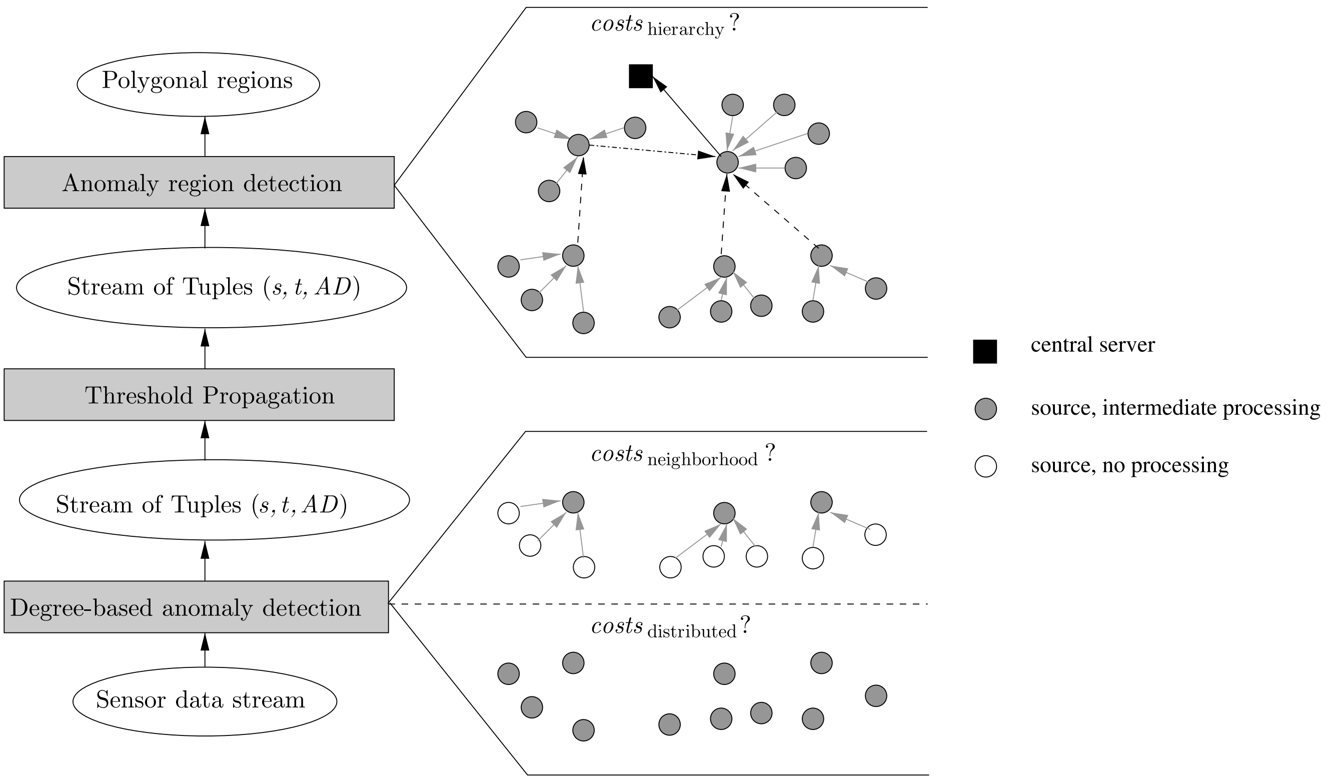


Fig. 5 Possibilities of in-network processing

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

Threshold propagation and anomaly region detection cannot be processed on the individual sensors or for each neighborhood independently, because we also have to detect regions that cross neighborhood boundaries. 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 of 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 cost C (in $\mu J/s = W$ (Watt)) are:

- c_{msg}: constant cost for a single message (header etc.) in μJ
- c_{byte}: additional cost for each byte in a message in μJ
- $c_{cpu}(op)$: cost for processing operation op on a node in μJ
- r_m: the rate of taking measurements in $1/s$
- r_a: the rate of events, i.e., the average rate an anomaly is detected, in $1/s$
- 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 the shortest paths in the node hierarchy)

For our cost model, we assume that nodes are organized in a balanced binary tree.

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

5.1 Central computation

Independent of the degree of distributed computing used, each node samples its data periodically. The cost C^{sampl} for

Table 1 Power consumption on real sensors

Measurement	Time	Energy
MCU wake up	115 µs	0.416 µJ
Sample humidity	71 ms	75.92 µJ
Sample temperature	221 ms	270.92 µJ
Sample light	18 ms	23.4 µJ
Sample CO_2	0.5 s	237.5 µJ

the sampling are dependent of the used sampling rate r_m, the microcontroller unit (MCU) wake up cost c_{wake} and the considered physical sensor measurement cost $c_{measure}$:

$$C^{sampl} = r_m \cdot c_{wake} \cdot c_{measure} \cdot m$$

Please note that the used sampling rate r_m dependents on the used sensor type. Table 1 shows measurements for power consumptions of different real sensors. The example shows that a measurement of CO_2 is more expensive than the measurement of humidity or light. Additionally one can see that the time necessary for a single measurement differs significantly such that the sampling rate for instance for CO_2 should be more than for light or humidity.

In the centralized approach, we consider the cost for transmitting data. The cost for processing at the central server ist 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). A single measurement can be expressed using 2 bytes. Thus, we obtain the following cost for sending data in our central scenario

$$C^{send}_{centr} = h \cdot r_m \cdot (c_{msg} + 2 \cdot c_{byte}) \cdot m$$

Receiving a message results in energy consumption as well. In our experiments, we observed that this is about the same cost as sending a message; see Sect. 6 for more details on this. For receiving data, we obtain the following cost

$$C^{recv}_{centr} = (h-1) \cdot r_m \cdot (c_{wake} + c_{msg} + 2 \cdot c_{byte}) \cdot m$$

With this, the total cost for data measurement and transmission in the central approach is as follows:

$$C_{centr} = C^{sampl} + C^{send}_{centr} + C^{recv}_{centr}$$

If we assume a single-hop scenario where each source sends data in a direct manner to the central instance, that is $C^{recv}_{centr} = 0$ and $h = 1$, the resulting total cost is

$$C_{single} = C^{sampl} + r_m \cdot (c_{msg} + 2 \cdot c_{byte}) \cdot m$$

Even if there are 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 messages that need to be resent. Furthermore, we assume that 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.

5.2 Distributed anomaly detection

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 burst detection and outlier detection approaches discussed earlier in the case no information about the neighborhood is involved.

Processing data on source nodes results in additional computing cost $c_{cpu}(process)$, which can differ depending on the used algorithm and its parameters. Since each measured item must be processed, we get the following computing cost

$$C^{process}_{*} = r_m \cdot c_{cpu}(process) \cdot m$$

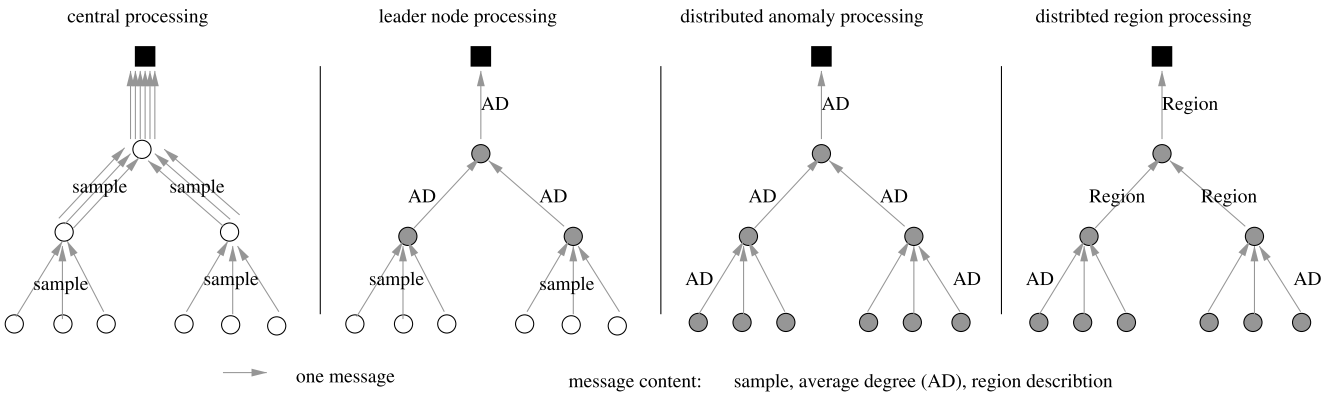


Fig. 6 Message propagation in the different in-network processing approaches

$C_*^{process}$ denotes the processing cost. In the following formulas $*$ is replaced by the according algorithm identifier.

In general, processing data will be done immediately after the measurement, that is, there is no additional wake up cost. The selectivity of the anomaly detection should be less than 1 (just in this case an in-network processing is reasonable). In our anomaly detection example we have a selectivity of $\sigma = r_a/r_m$, where r_a denotes the average rate an anomaly is detected. Thus, we obtain the following cost for sending and receiving data:

$$C_{anomaly}^{forwAD} = \underbrace{h \cdot r_a \cdot (c_{msg} + 2 \cdot c_{byte}) \cdot m}_{send\ anomalies} + \underbrace{(h-1) \cdot r_a \cdot (c_{wake} + c_{msg} + 2 \cdot c_{byte}) \cdot m}_{receive\ anomalies}$$

The overall cost for anomaly detection on individual sensors thus is

$$C_{anomaly} = C^{sampl} + C_{anomaly}^{process} + C_{anomaly}^{forwAD}$$

Obviously, this can only help to reduce energy consumption if r_a is significantly less than r_m, that is $\sigma \ll 1$. This should be the usual case, as we are dealing with anomalies rather than normal situations. For most sensors, c_{cpu} is orders of magnitude lower than c_{msg}.

5.3 Neighborhood based anomaly detection

If we take information about neighboring sensors 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. The cost for sending and receiving a measurement is

$$C_{lead}^{forw} = \underbrace{r_m \cdot (c_{msg} + 2 \cdot c_{byte}) \cdot (m - m_l)}_{send\ measurements} + \underbrace{r_m \cdot (c_{wake} + c_{msg} + 2 \cdot c_{byte}) \cdot (m - m_l)}_{receive\ measurements}$$

In this approach leader nodes only send the resulting anomaly degree to nodes higher in the routing tree. The resulting communication cost thus is

$$C_{lead}^{forwAD} = \underbrace{(h-1) \cdot r_a \cdot (c_{msg} \cdot m_l + 2 \cdot c_{byte} \cdot m)}_{send\ anomalies} + \underbrace{(h-2) \cdot r_a \cdot ((c_{wake} + c_{msg}) \cdot m_l + 2 \cdot c_{byte} \cdot m)}_{receive\ anomalies}$$

Therefore, we obtain the following cost for the distributed neighborhood based anomaly detection:

$$C_{lead} = C^{sampl} + C_{anomaly}^{process} + C_{lead}^{forw} + C_{lead}^{forwAD}$$

As this assumes uniform distribution of all $r_a \cdot m$ anomalies over the m_l leader nodes, the cost C_{lead} represents 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.

5.4 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 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 at 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

```
   Input: $\varphi$ [, 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

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.

We assume, that source nodes send the anomaly degree to higher nodes within the routing tree in a direct manner, that is, no additional forwarding is necessary. The one hop cost for the anomaly forwarding is

$$C_{region}^{forwAD} = \underbrace{r_a \cdot (c_{msg} + 2 \cdot c_{byte}) \cdot (m - m_l)}_{send\ anomalies} + \underbrace{r_a \cdot (c_{wake} + c_{msg} + 2 \cdot c_{byte}) \cdot (m - m_l)}_{receive\ anomalies}$$

Processing regions is only handled by higher nodes within the network hierarchy and will be done only if anomalies are received, that is nodes have received an anomaly before and are already in a wake up mode.

$$C_{region}^{updReg} = r_a \cdot L \cdot c_{cpu}(process) \cdot m_{aR}$$

After the successful region computation, the region information is propagated within the sensor network. We assume, that only those nodes wake up that can handle the propagated regions (i.e. using multiple-frequency operations [12]). With this assumption we get for the region propagation the following cost

$$C_{region}^{propReg} = \underbrace{r_a \cdot L \cdot (c_{msg} + size_{\{\mathcal{O},\mathcal{P},\mathcal{I}\}} \cdot c_{byte}) \cdot m_{lR}}_{send\ regions} + \underbrace{r_a \cdot L \cdot (c_{wake} + c_{msg} + size_{\{\mathcal{O},\mathcal{P},\mathcal{I}\}} \cdot c_{byte}) \cdot m_{lR}}_{receive\ regions}$$

Forwarding finished regions to the central node results in the following cost

$$C_{region}^{forwReg} = \underbrace{r_a \cdot h \cdot (c_{msg} + size_R \cdot c_{byte}) \cdot m_{fR}}_{send\ regions} + \underbrace{r_a \cdot (h-1) \cdot (c_{msg} + size_R \cdot c_{byte}) \cdot m_{fR}}_{receive\ regions}$$

Based on these assumptions and the algorithm described above, we obtain the total cost for the in-network distributed region processing

$$C_{region} = C^{sampl} + C^{process}_{anomaly} + C^{forwAD}_{region} + C^{updReg}_{region} + C^{propReg}_{region} + C^{forwReg}_{region}$$

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 cost of the chosen method. Note that all listed cost formulas are worst case approximations, as we use average values etc. Nevertheless, they are suited for analytically evaluating in-network processing by comparing the cost of each option. This is done in the following Sect. 6.

6 Evaluation

In the following, we focus on the evaluation of the distributed version of our approach. The centralized region detection approach has been evaluated in [10], and we showed the feasibility of the threshold propagation in Sect. 4. Here we now present an analytical evaluation of the in-network processing options introduced in Sect. 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 to (1) identify the sensitive factors that have most influence on the actual choice on the degree of distribution, and to (2) determine the benefits one gets from in-network processing and in which situations such benefits occur. We expect the in-network methods to be 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 MCU MSP430F1611, 4 MHz clock rate, IEEE 802.15.4 compatible CC2420 transceiver with 250 kBit/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 0.384 ms. Consequently, with 10 bytes raw data there are 21 bytes sent (about 0.672 ms). Table 2 summarizes the most important results of the tests.

Roughly speaking, local processing is about 1000 times cheaper than communication. This factor has also been identified in several other works, such as [6]. However, 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. This would not apply if the TDMA protocol was used instead – 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. Furthermore, 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 Sect. 5 with concrete values. For this, we derived the average values from Table 2. 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 is clearly r_a. This is illustrated in Fig. 7. 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

Table 2 Average energy consumption measured on real sensors

Measurement	Time	Energy
Compute average of 10 values	52.3 µs	0.272 µJ
Compute average of 100 values	245 µs	1.274 µJ
Single addition	2 µs	0.010 µJ
Single division	27 µs	0.140 µJ
Single multiplication	16.2 µs	0.08 µJ
Sending 1 byte	4.85 ms (2.33–6.95 ms)	240.19 µJ (121–361 µJ)
Sending 10 bytes	4.9 ms (2.8–7.4 ms)	252.93 µJ (146–385 µJ)

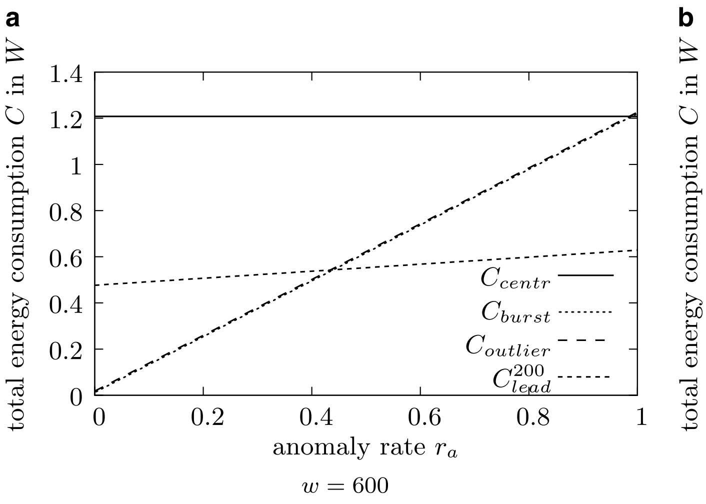


Fig. 7 Varying anomaly rate r_a ($m = 1000$, $m_l = 200(150)$, $h = 3$, $r_m = 1$)

$C_{outlier}$, respectively. The anomaly detection method using leader nodes is referred to by C_{lead}. Figure 7a shows that message costs significantly outweigh processing costs. 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 Fig. 7b show that all methods scale with the window size w, which is the size of the sliding window on which anomaly detection is done (see Sect. 2.1). Only the cost of the method using leader nodes is significantly affected by w. With $m_l = 200$ leader nodes the cost of the in-network method is lower than the cost of the central approach, up to an anomaly rate of $r_a = 0.5$ (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, as shown in Fig. 8. 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 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 values of w.

Figure 8 also shows the rather weak effect of increasing the depth of the hierarchy, i.e., the average hop count. In contrast to Fig. 7 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 Fig. 7b. In Fig. 8 the energy consumption for $w = 5000$ is at 1.17, whereas the central approach increased by about 0.5 Watts.

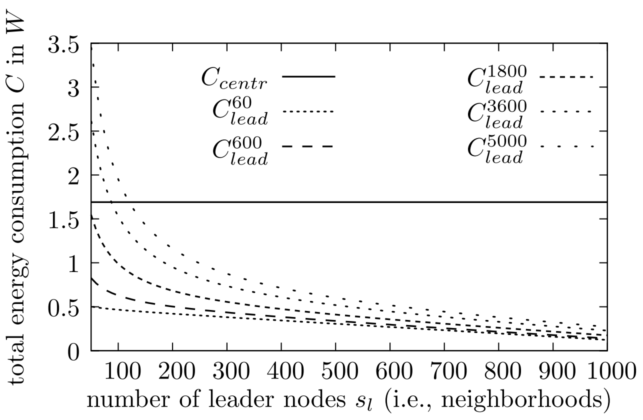


Fig. 8 Varying number of leader nodes m_l ($m = 1000$, $h = 4$, $r_m = 1$, $r_a = 0.1$)

Finally, we evaluated the in-network processing of anomaly region detection. As this is based on anomaly detection, we determined the cost for both steps in conjunction (as we already did in the formula in Sect. 5). This only concerns the CPU costs for each method and influences performance of region detection negligible. 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 con-

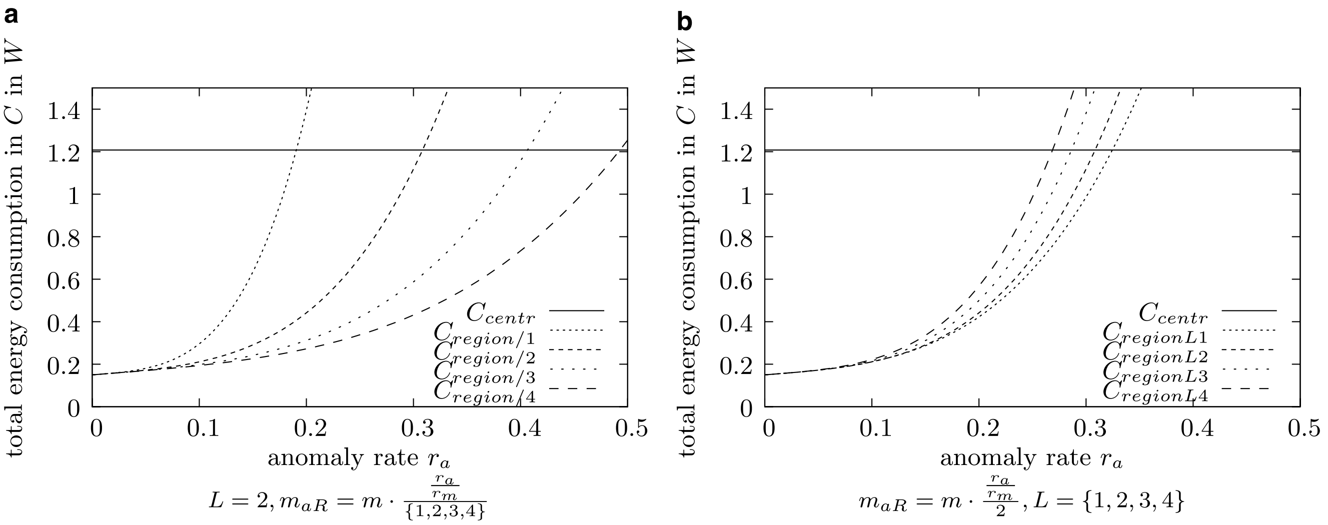


Fig. 9 In-network region detection ($m = 1000$, $m_l = 200$, $h = 3$, $r_m = 1$, $w = 5000$)

cluded that the most influencing parameters are m_{aR} and L. In Fig. 9 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 9 shows that, as expected, energy consumption for in-network region detection is much higher than for just 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. It can be seen in Fig. 9a that the anomaly rates where in-network processing is worthwhile become smaller as the number of regions increases (larger m_{aR}). Figure 9a also shows that the more regions occur, the larger is the increase of energy consumption. 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 Fig. 9b).

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 (i.e. a pc with a data stream engine). But in the case of low anomaly rates it is still a good option for saving energy to perform the detection in-network. The benefit of in-network region detection decreases with increasing number of regions and size of these regions – due to the hierarchy-based approach.

7 Conclusion and outlook

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 obstacles in a given sensor field. The presented algorithm allows us to derive anomaly regions with meaningful boundaries instead of regions described only by grouping of 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.

As part of our ongoing work we plan to extend the framework for supporting further anomaly detection techniques. This includes other outlier models as well as support for high-dimensional sensor data. This will facilitate region detection in projected space, such that the detected regions indicate in which spatial areas certain dimensions of the feature space are correlated. A second line of research is to consider physical propagation models of the observed phenomena. If we know how an event diffuses through a region and along or around obstacles we could improve the accuracy of the boundary placement. Finally, we will integrate the anomly detection algorithm into our data stream processing system AnduIN[1], which already supports distributed sensor-local processing.

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

References

1. MIT Computer Science and Artificial Intelligence Lab (2004) Intel lab sensor data. http://db.csail.mit.edu/labdata/labdata.html
2. California Irrigation Management Information System (CIMIS) (2008) http://wwwcimis.water.ca.gov
3. Angiulli F, Fassetti F (2007) Detecting distance-based outliers in streams of data. In: Proc. 16th ACM Conference on Conference on Information and Knowledge Management (CIKM'07), New York, pp. 811–820
4. Basu S, Meckesheimer M (2007) Automatic outlier detection for time series: an application to sensor data. Knowl Inf Sys 11(2):137–154
5. Kant Chintalapudi K, Govindan R (2003) Localized edge detection in sensor fields. In: Proc. 1st IEEE Int. Workshop on Sensor Network Protocols and Applications, May 2003, pp. 59–70
6. Culler D, Estrin D, Srivastava M (2004) Overview of sensor networks. IEEE Comput 37(8):41–49
7. Singh Dhillon S, Chakrabarty K (2003) Sensor placement for effective coverage and surveillance in distributed sensor networks. In: Proc. of IEEE Wireless Communications and Networking Conference, pp. 1609–1614
8. Ding M, Chen D, Xing K, Cheng X (2005) Localized fault-tolerant event boundary detection in sensor networks. In: Proc. of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies INFOCOM 2005, March 2005, Vol. 2, pp. 902–913
9. Estivill-Castro V, Lee I (2001) Fast spatial clustering with different metrics and in the presence of obstacles. In: Proc. 9th ACM Int. Symposium on Advances in Geographic Information Systems (ACM-GIS'01), Atlanta, pp. 142–147
10. Franke C, Gertz M (2008) Detection and exploration of outlier regions in sensor data streams. In Proc. Int. Workshop on Spatial and Spatiotemporal Data Mining (SSTDM'08), Dec 2008
11. Gehrke J, Madden S (2004) Query processing in sensor networks. IEEE Pervas Comput 3(1):46–55
12. Gu L, Stankovic JA (2005) Radio-triggered wake-up for wireless sensor networks. Real-time Sys 29:157–182
13. Han J, Kamber M, Tung AKH (2001) Spatial Clustering Methods in Data Mining: A Survey. In: Miller H, Han J (eds) *Geographic Data Mining and Knowledge Discovery*. CRC Press, pp. 201–230
14. Klan D, Karnstedt M, Pölitz C, Sattler KU (2008) Towards burst detection for non-stationary stream data. In: Proc. Workshop on Knowledge Discovery, Data Mining, and Machine Learning (KDML 2008)
15. Kolingerová I, Zalik B (2006) Reconstructing domain boundaries within a given set of points, using delaunay triangulation. Comput Geosci 32(9):1310–1319
16. Krause A, Guestrin C, Gupta A, Kleinberg J (2006) Near-optimal sensor placements: maximizing information while minimizing communication cost. In: Proc. of the 5th Int. Conference on Information Processing in Sensor Networks (IPSN '06), New York, pp. 2–10
17. Krishnamachari B, Estrin D, Wicker SB (2002) The impact of data aggregation in wireless sensor networks. In: ICDCSW '02, Washington, DC, pp. 575–578
18. Lu C-T, Kou Y, Zhao J, Chen L (2007) Detecting and tracking regional outliers in meteorological data. Inf Sci 177(7):1609–1632
19. Nowak R, Mitra U (2003) Boundary estimation in sensor networks: Theory and methods. Inform Proc Sens Netw 2634:80–95
20. Shasha D, Zhu Y (2004) *High Performance Discovery in Time Series: Techniques and Case Studies*. Springer, Berlin
21. Subramaniam S, Palpanas T, Papadopoulos D, Kalogeraki V, Gunopulos D (2006) Online outlier detection in sensor data using non-parametric models. In: Proc. Int. Conf. on Very Large Data Bases (VLDB'06), pp. 187–198
22. Tung AKH, Hou J, Han J (2001) Spatial clustering in the presence of obstacles. In: Proc. 17th Int. Conference on Data Engineering (ICDE'01), Washington, DC, IEEE Computer Society, pp. 359–367
23. Wu W, Cheng X, Ding M, Xing K, Liu F, Deng P (2007) Localized outlying and boundary data detection in sensor networks. IEEE Trans Knowl Data Engin 19(8):1145–1157
24. Yao Y, Gehrke J (2003) Query processing for sensor networks. In: Proc. of the 1st Biennial Conference on Innovative Data Systems Research (CIDR'03), January 2003
25. Yao Y, Gehrke J (2002) The cougar approach to in-network query processing in sensor networks. ACM SIGMOD Record 31(3):9–18
26. Zhang J, Lou M, Ling TW, Wang H (2004) Hos-miner: a system for detecting outlying subspaces of high-dimensional data. In: Proc. Int. Conf. on Very Large Data Bases (VLDB'04), VLDB Endowment, pp. 1265–1268
27. Zhang J, Papadias D, Mouratidis K, Zhu M (2004) Spatial queries in the presence of obstacles. In: Proc. 9th Int. Conference on Extending Database Technology (EDBT'04), Heraklion, Crete, Greece, pp. 366–384
28. Zhang X, Shasha D (2006) Better burst detection. In: Proc. 22nd Int. Conference on Data Engineering (ICDE'06), pp. 146–149
29. Zhao F, Liu J, Liu J, Guibas L, Reich J (2003) Collaborative signal and information processing: An information directed approach. Proc IEEE 1199–1209
30. Zhu Y, Shasha D (2003) Efficient elastic burst detection in data streams. In: KDD '03, New York, pp. 336–345

[1] http://www.tu-ilmenau.de/fakia/AnduIN.8598.0.html

Conny Franke is a Ph.D. student in the Department of Computer Science at the University of California, Davis. She received her Diploma (M.Sc.) in Computer Science from the Ilmenau University of Technology, Germany, in 2005. Her current research interests include data mining on data streams, spatial, and spatio-temporal data as well as resource-adaptive and quality-aware stream mining.

Marcel Karnstedt received his Diploma (M.Sc.) from the Martin-Luther-Universität Halle-Wittenberg, Germany, at the end of 2003. From January 2004 to February 2009 he was employed as research associate and teaching assistant in the Databases & Information Systems Group at the Ilmenau University of Technology, Germany. Since March 2009 he is member of the Information Mining and Retrieval Unit at the Digital Enterprise Research Institute, National University of Ireland, Galway. His current research interests are in large-scale graph mining (particularly social network graphs), P2P databases, and resource-adaptive and quality-aware stream mining.

Daniel Klan is a Ph.D. student at the Database & Information Systems group at the Faculty of Computer Science and Automation of the Ilmenau University of Technology, Germany. He received his Diploma (M.Sc.) in Computer Science from the University of Jena, Germany. His current research interests include data mining on data streams and sensor networks.

Michael Gertz is a full professor at the University of Heidelberg where he heads the database systems group at the faculty of Mathematics and Computer Science. He received his diploma in Computer Science from the University of Dortmund, Germany, in 1991 and his Ph.D. from the University of Hannover, Germany, in 1996. From 1997 until 2008 he was a faculty at the University of California at Davis. He currently serves on the editorial board of the VLDB Journal, and he is an associate editor of the ACM Journal on Data and Information Quality. His research interests include the management and analysis of scientific data, with a particular focus on streaming, spatial and spatio-temporal data.

Kai-Uwe Sattler is full professor and heads the Database and Information Systems group at the Faculty of Computer Science and Automation of the Ilmenau University of Technology, Germany. He received his Diploma (M.Sc.) in Computer Science from the University of Magdeburg, Germany. He received his Ph.D. in Computer Science in 1998 and his Habilitation (venia legendi) in Computer Science in 2003 from the same university. He has published five textbooks and more than 100 research papers. His current research interests include autonomic features in database systems and large-scale distributed data management.

Elena Chervakova received the B.Sc. degree in computer engineering from the Moscow Power Engineering Institute (Technical University), Moscow, Russland, the M.Sc. degree in computer engineering from Ilmenau University of Technology, Germany. She is currently working toward the Ph.D. degree in electrical and computer engineering at the Ilmenau University of Technology and is a researcher at the Institute of Microelectronics- and Mechatronics Systems, Ilmenau, Germany. Her research interests include signal processing, control and automation, wireless sensor networks and localization.