Managing Evolving Shapes in Sensor Networks

Besim Avci
Dept. of EECS
Northwestern University
Evanston, Il

Goce Trajcevski*
Dept. of EECS
Northwestern University
Evanston, Il

Peter Scheuermann
Dept. of EECS
Northwestern University
Evanston, Il

besim@u, goce@eecs, peters@eecs.northwestern.edu

ABSTRACT

This work addresses the problem of efficient distributed detection and tracking of mobile and evolving/deformable spatial shapes in Wireless Sensor Networks (WSN). The shapes correspond to contiguous regions bounding the locations of sensors in which the readings of the sensors satisfy a particular threshold-based criterion related to the values of a physical phenomenon that they measure. We formalize the predicates representing the shapes in such settings and present detection algorithms. In addition, we provide a light-weight protocol and aggregation methods for energy-efficient distributed execution of those algorithms. Another contribution of this work is that we developed efficient techniques for detecting a co-occurrence of shapes within a given proximity from each other. Our experiments demonstrate that, when compared to the centralized techniques – which is, predicates being detected in a dedicated sink – as well as distributed periodic contours construction, our methodologies yield significant energy/communication savings.

Categories and Subject Descriptors

H.0 [**Information Systems**]: General

General Terms

Algorithm, Design

Keywords

Spatio-temporal events, Wireless Sensor Networks, WSN, Evolving shapes

1. INTRODUCTION

Wireless Sensor Networks (WSN) consist of a large number (hundreds, or even thousands) of tiny devices which, in addition to the capabilities to sense/measure values of a particular physical phenomena and perform basic calculations, can self-organize in a wireless network [3]. This ability, in turn, enables communicating various observations from nodes located in different parts of the network, which has spurred WSNs as paradigm of choice in a plethora of applications: scientific observations and actuations, traffic management, environmental safety/hazards detections, infrastructure monitoring, health-care and military surveillance [15, 26, 30] – to name but a few.

For many typical sensor nodes, it is a "fact of life" that the communication – be it a transmission or active listening/reception – drains a 2-3 orders of magnitude more energy than the processes associated with sensing and local-computations tasks [3]. Given that the batteries on many types of sensors are non-renewable, especially when deployed in inaccessible terrain, avoiding unnecessary communication is a paramount.

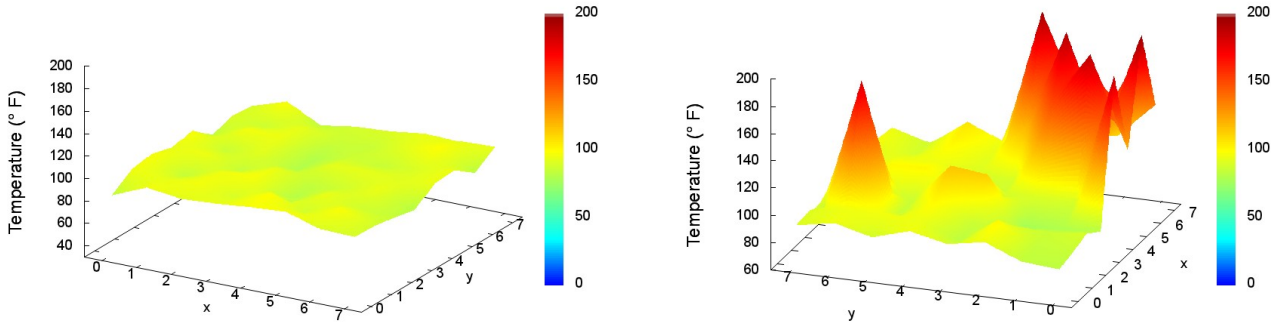
A common property of many of the phenomena monitored by WSNs is that the data sources may span over large spatial regions and generate values at distant (discrete) locations. However, in many applications, it is of a great importance to detect the spatial boundaries of the (sub)regions for which the sensed values have a particular property (e.g., temperature exceeding certain threshold). Several research works have addressed different aspects of the problem of detecting and representing spatial features of a particular phenomenon [13, 33, 12]. As an example, to enable energy-efficient querying, spatial summaries – e.g., isocontours [13]– may be constructed, representing the (boundaries of the) regions with same (or, close-enough) readings. A natural trade-off is the precision of the aggregated representation vs. the energy efficiency.

Our goal in this work is to provide distributed algorithms that will enable energy efficient in-network detection of evolving spatial shapes – which is, contiguous 2D regions in which readings of the sensors satisfy a particular threshold-based property. What motivates this work is the observation that due to the changes of the values of the monitored phenomena, the boundaries of the spatial regions (i.e., shapes) with homogeneous readings may change over time and, based on those changes, actions with potentially costly consequences may need to be taken. For example, if the region(s) with high-temperature indicates a forest fire, knowing how it evolves is a paramount for planning the distribution of the fire-fighters equipment, as well as predicting

*Research supported in part by NSF grants CNS-0910952 and III 1213038, and ONR grant N00014-14-1-0215.

SSDBM '14, June 30 - July 02 2014, Aalborg, Denmark

ACM 978-1-4503-2722-0/14/06 ...$15.00
http://dx.doi.org/10.1145/2618243.2618264



(a) Normal conditions

(b) Abnormal readings

Figure 1: Detecting contiguous evolving shapes

where herds of animals may be escaping. Similarly, in other scenarios related to toxic fumes spreading, the manner of evolution may help in selecting evacuation routes with minimal risks. Among the challenges is the lack of formal treatment of the different facets of this problem. As an illustration, having very few scattered spikes of temperature readings may be due to noise; similarly, a large number of small fluctuations need not indicate an occurrence of something of interest. However, when the set of temperature readings is very high and the detecting sensors are spatially close and cover a large-enough area, it may be an indication of a situation which may require a certain actuation – e.g., from turning sprinklers in a field, to mobilizing fire-fighters due to forest fire. An illustrating scenario[1] for our objective is presented in Figure 1. Figure 1(a) shows a "normal" scenario in which the readings of the sensors deployed in the field are below the threshold of 100°C, which indicates that there is no danger of a fire. However, in Figure 1(b) we have high-readings of some sensors, prompting two complementary observations: (1) The reading in the left part of Figure 1(b) may indicate an "outlier" (i.e., a malfunctioning sensor); (2) The readings in the right portion of Figure 1(b) not only indicate a potential hazard (e.g., a forest-fire) but also demand some description of the spatial boundary of the the shape in which such values have been read. Clearly, once the validity of the corresponding shape-describing predicate has been established, the next task is to efficiently track/-monitor how such shape(s) evolves over time.

In addition to the detection and monitoring of the boundaries of evolving shapes, in this work we also consider the problem of efficient detection of co-occurrence of spatial shapes – which is, spatial shapes which are closer than a certain distance-threshold. While there have been research efforts addressing spatial summaries construction and maintenance [13, 33] as well as shapes interpolations [28], to the best of our knowledge, there has been no attempts to efficiently detect co-occurrence of such spatial phenomena.

[1]The images are actually generated from the phenomena fluctuations that we used in the SIDnet-SWANS simulator [14] in our experiments (cf. Section 5).

Such problems, however, are of extreme importance in scenarios like disaster mitigation. If we consider again the forest fire scenario, there can be two origins for forest fire and these fire regions may move/expand over time – eventually they may even merge. The information of two regions coming closer in this example carries a great deal of importance when it comes to both controlling the fire by deploying firemen, as well as the escape-routes affecting the safety of the firemen.

Towards these goals, the main contributions of this work are developing an approach for efficient in-network detection and tracking of evolving spatial shapes, along with detection of the co-occurrence of those shapes which are within a proximity-threshold. We formalize the notion of a spatial shape and provide efficient distributed algorithms for processing the corresponding predicates in WSNs. Our experiments indicate that the methods we propose are much more efficient than the centralized (naïve) scheme, which would transmit all the individual observations' data to a dedicated sink and have the shape construction performed there. In addition, the event-based shapes detection approach is demonstrated to perform better than the traditional approach of periodic shape-construction.

The rest of this paper is organized as follows. After recollecting basic preliminaries and developing the terminology in Section 2, we proceed with the details of the shape detection methodology in Section 3. Section 4 expands towards the details of the protocol and distributed algorithms for detecting co-occurrence of evolving shapes. Our experimental observations are discussed in Section 5. Section 6 positions our results with respect to related literature and Section 7 concludes the paper and outlines directions for future work.

2. BACKGROUND AND PRELIMINARIES

A wireless sensor network is assumed to consist of N nodes, represented as the set $SN = \{sn_1, sn_2, \ldots, sn_N\}$. Each node sn_i, in addition to measuring values of the phenomena in its vicinity and communicating with its neighbors, is also aware of its location (x_i, y_i) in a corresponding reference

coordinate system – either via GPS or other collaborative techniques [3]. Energy efficiency in WSN applications has been known to benefit from event-based processing, as opposed to periodic sampling and transmission, in a variety of scenarios. Events can be defined using constructs of the available query languages such as TinySQL [21] – or they can also be hard-coded when programming each individual mote.

In the rest of this section, we firstly present an overview of the impact of the event-based processing in different aspect of WSNs systems – and show how we utilize the existing findings in our work. Subsequently, we follow with presenting the formal definition of the predicates introduced in this work.

2.1 WSN – Broad Perspective

Aside from the context of a particular application settings – from a "generic" perspective, routing structures and duty cycling are some of the important factors of event based processing in WSNs. However, there are other roles of the event-based modelling and executing of tasks in WSNs.

Routing and Aggregation

Routing protocols in WSN can be classified in two basic categories: hierarchical and flat [2]. Flat routing schemes include methods like flooding – and the commonality of such methods is that typically no particular topology is required. Hierarchical routing protocols [18], in turn, can also be divided into two sub-categories: tree-based and cluster-based. Tree-based protocols, as the name suggests, form a tree with all the nodes rooted at the sink and data propagation is achieved by relaying messages from children to parents. Cluster-based protocols divide the nodes into groups which are based on a local proximity within a particular geographic region. Each such group is managed by a designated cluster head.

Data gathering and aggregation in WSNs is often done with an aid of a hierarchical indexing structure. The routing protocol that we rely upon in our shape detection approaches is based on splitting the zone of interest (i.e., the sensing field) into nxn grid of cells with equal areas. Each such cell is assigned a local *principal*[2], which is in charge of gathering the data from all the nodes inside the region of that cell. We note that other space partitions may be considered (e.g, K-d tree [23]) – based on the number of sensors within a cell (possibly accompanied with a limit on the number of hops when communicating with the principal). However, the important observation is that the principals are subsequently organized in a tree, rooted in a dedicated sink. This geographic clustering has a two-fold impact: (1) it helps in localizing the decision-making because each principal can detect shape within its corresponding cell; and (2) the tree/hierarchy principals enables a distributed solution not only for the problem of detecting the (co)occurrence of shapes, but also a single shape spanning across ≥ 2 cells. Propagation of the messages can be briefly expressed as: firstly data is propagated from sensor nodes to the cell/region principals; subsequently, the principals send the data up their own hierarchy, attempting to limit the level of traversal (or, avoiding a transmission all the way up to the sink).

[2]We use the term "principal", although in WSN literature they are often referred to as "local cluster heads". However, we do so in order to avoid the ambiguity with the more traditional concept of clustering (cf. [10]).

Periodic vs. Event-driven Sampling

WSNs typically apply duty-cycling for their nodes in order to save energy by having the nodes wake up periodically [5]. One possibility for duty-cycling is to use cluster-based routing protocols for applying Time-Division Multiple Access (TDMA) for the nodes in the jurisdiction of a particular cluster head, making the nodes send their sensed values within their own allocated time slot. Periodic data transmission and event-driven transmission mainly differ in terms of when they send sensed value. In periodic schemes, every sensor sends their sensed value to their respective cluster head in synchronized periods (e.g., epochs in TinyDB [21]). However, in an event-driven scheme the data transmission depends on how a given query is specified beforehand. For instance, data may be transmitted whenever a particular sensed value reaches/exceeds a threshold. Doing so may yield substantial energy savings when the fluctuations of the monitored phenomenon are not too frequent.

Clearly, such behavior of a WSN needs a well-defined collection of events of interest (e.g., a temperature of a patient exceeding 38°C). We note that detection of events may also be performed cooperatively by multiple sensors like, for example in applications dealing with location-detection and tracking where a distance from an object should be sensed by at least 3 different sensors to pinpoint the estimate location of the object [19, 32].

– *TinyDB and TinySQL* WSNs can also be perceived as distributed databases which are capable of processing continuous and instantaneous queries [31]. As their names suggest, instantaneous queries are aim at capturing values in a single time-instant (which, in some sense can be thought of as instantaneous events), whereas continuous queries are more convenient for applications which require periodic sampling and may need comparison of "historic" states. TinyDB [21] is a system tailored for database applications in WSN context. It supports both continuous and instantaneous queries and it provides and interface through which the users can specify queries and events of interests using the SQL-like syntax.

2.2 Shape-detection Predicates

In many regards, the predicates that we focus upon in this work can be perceived as *events* – i.e., occurrence of "something of interest" – which, in turn, are commonly split into two basic categories: primitive and composite [1]. We note that throughout this work, when it comes to primitive events such as readings of individual sensors, we do not distinguish between the *detection* and the actual *occurrence*. Typically, the primitive events are associated with values which are directly measurable or observable – like, for example, a sensor reading above a certain threshold – whereas the composite ones are specified using well-defined event algebras (both in terms of the syntax and semantics/processing) [16].

We note that throughout this work we concentrate on shapes represented by simple polygons – which is, no holes and no

intersecting edges. In this spirit, our first composite event is the one which denotes an *occurrence of a shape which satisfies "certain properties"* – where the properties are specified as parameters of the event.

DEFINITION 1. *Given a wireless sensor network SN, the predicate $E_S(\gamma, A, t)$ holds in SN iff there exists a connected spatial shape (S) such that:*

1. *The area of S is $\geq A$.*
2. *The smallest value read by every sensor inside S is at least γ.*
3. *The time of detection of E_S is t.*

We note that other parameters may be added – depending on particular settings or application's demands. For example, if one would like to provide a bound on how "narrow" a particular area can get, the argument signature may be augmented as $E_S(\gamma, r, A, t)$. This will limit the set of qualified shapes only to the ones for which the diameter of the point-set corresponding to the sensors locations is $\leq r$.

Our next predicate denotes a co-occurrence of two spatial shapes which, in addition to each of them satisfying 1, are also within certain distance from each other and have been detected within an acceptable time-interval. More formally:

DEFINITION 2. *Given a wireless sensor network SN, the predicate $E_{CO}(E_{S1}, E_{S2}, d, \delta)$ holds in SN iff*

1. *$E_{S1}(\gamma_1, r_1, A_1, t_1)$ holds in SN.*
2. *$E_{S2}(\gamma_2, r_2, A_2, t_2)$ holds in SN.*
3. *$|t_1 - t_2| \leq \delta$*
4. *The Euclidian distance between the two closest points of the boundaries of A_1 and A_2 is $\leq d$.*

In this work, we only consider the Euclidean distance, although other distance functions may be applicable (e.g., Hausdorff distance or the distance between the centroids).

We conclude this section with a note that the arguments signature in the definition of the predicates may be subject to minor variations – for the purpose of enabling more efficient detection. We will discuss this issue in greater detail in the subsequent sections.

3. SHAPE DETECTION METHODOLOGY

We now present the main aspect of the procedural semantics – which is, the data structures and the algorithms – used to detect the occurrence of $E_S(\gamma, A, t)$. We distinguish two separate facets of the problem: (1) Instantaneous detection of the predicate; (2) Continuous monitoring upon initial detection. In the sequel, we address each of them in greater detail.

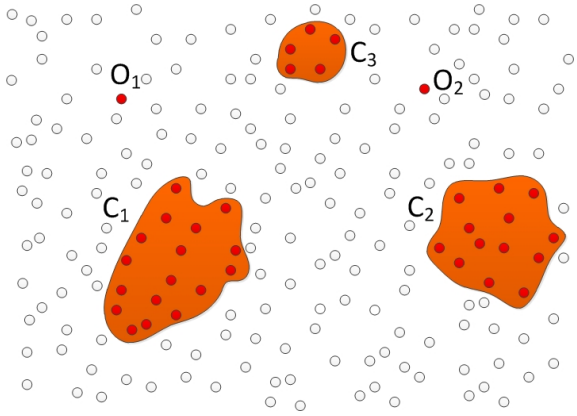


Figure 2: Cluster Head Snapshot

3.1 Initial Shape Detection

Upon deployment and dissemination of the request, the principal node of each region begins to receive the readings from the sensors in its region which exceed the value of γ, along with the locations of the corresponding sensors. Hence, the principal has a collection of all the (x_i, y_i, val_i) readings, specifying the location of each sensor for which the value $val_i \geq \gamma$). In Figure 2 this is illustrated with the red filled disks, indicating that the sensors in the corresponding locations have readings above the given threshold.

Based on the value of the argument A in the specification of E_S, it may be the case that a particular not every sensor (or group of sensors) with reading above the threshold γ will not contribute towards detection of a shape with the desired features. Firstly, some of the values that the principal has received from a given location, may have been due to noise, random spikes in the phenomenon, or even communication - based error. This is illustrated in the top portion of Figure 2 with the 2 isolated measurements $\geq \gamma$, represented O_1 and O_2. Secondly, although multiple sensors close to each other may have readings $\geq \gamma$, it may be the case that the total area of the shape that they cover[3] is $< A$. An illustration of this is provided with the cluster of nodes denoted C_3 in Figure 2.

The illustration of an occurrence of our $E_S(\gamma, A, t)$ predicate is provided at the bottom part of Figure 2. Namely, we see the two spatial shapes, denoted C_1 and C_2, defined by the corresponding sets of near-by sensors with readings $\geq \gamma$. Each of them covers a large-enough area to be reported as satisfying the predicate $E_S(\gamma, A, t)$.

One of the main challenges now becomes how to construct the shape which can represent the set of discrete location-points. Clearly, some kind of an approximation of the (boundary of the) actual region will need to take place. Towards that, we have the following policy:

If two sensor nodes sn_i and sn_j with readings $\geq \gamma$ are located within distance $\leq \varepsilon$ from each other, then any geographical

[3]As we will formally define shortly, the areas of the shape is actually obtained as the area of its approximated boundary-polygon.

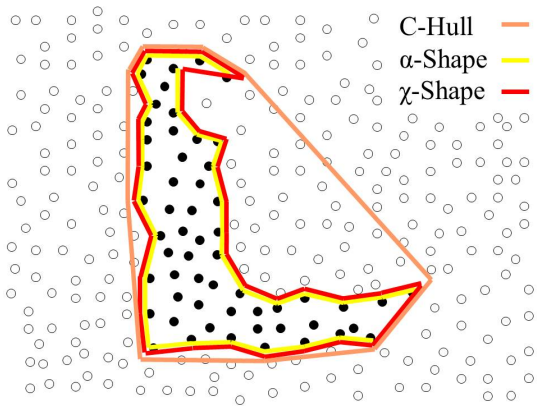


Figure 3: Convex Hull, α-shape, and χ-shape

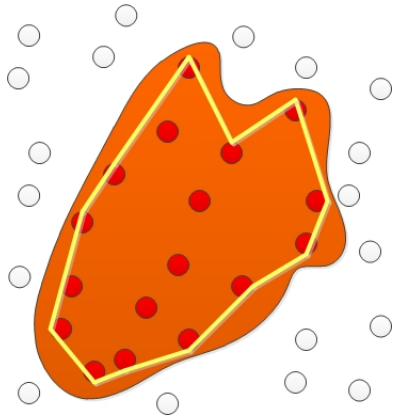


Figure 4: Region to Polygon

2D point within the union of the disks with radii ε and centered at the respective locations of sn_i and sn_j is assumed to satisfy the γ-threshold.

Essentially, we assume that the value of the sensed phenomenon is the same at any point "close enough" to the actual location of the sensor node which performed the reading. This, in turn, enables us to use bounding polygon construction techniques from set of sensor nodes' location.

Constructing a shape that will approximate the boundary of a discrete set of points is a problem studied in Computational Geometry and there are some popular approaches like, for example, convex hull of the set, α-shapes [8], etc. However, these techniques have certain shortcomings which render them not quite applicable in our contexts. As an example, consider Figure 3: (1) the convex hull of the $L-shaped$ collection of points looks creates a "pocket" within its interior, containing many points which do not belong to approximated set. (2) To soothe this "pocket-forming" kind of problems, α-shapes have been proposed – however, a drawback of the α-shapes is that they may result in disconnected regions in some cases, thus defeating the purpose of boundary construction.

For these reasons, in our work we rely on χ-shapes for representing the polygonal boundary of a set of points. In particular, we use an $O(n \log n)$ algorithm presented in [6]. A comparative illustration of these three approaches is shown in Figure 3.

In sum, given a collection locations of sensors with readings $\geq \gamma$ and within ε distance from each other, we construct χ-shapes for the set of location-points corresponding to those sensors. An important property of the boundary of the polygons generated as χ-shapes is that the polygon is simple, and the boundary is relatively more "accurate" with respect to characteristics of the outline of a shape. An illustration of the polygonal approximation of a region with this technique is shown in Figure 4 where the yellow line segment indicate the boundary of the point-set of the sensors shown.

Before we proceed with a formal presentation of our algorithm executed by the principal of a given region for the purpose of detecting the occurrences of the predicate E_s, we need to address one more issue. Namely, given the locations of all the sensor nodes with readings $\geq \gamma$, how do we detect which ones do we use as "generators" for separate χ-shape polygon? Towards this, we rely on the well-known DBScan algorithm for clustering [11]. Each cluster of 2D points is represented as a separate list, and all such lists are collectively kept as entries in a vector structure.

Formally, the procedure for detecting spatial shapes which satisfy the predicate $E_S(\gamma, A, t)$ is specified in Algorithm 1.

We maintain two sequential structures (vectors): ES, storing the (locations of the) detected events with readings $> \gamma$; and VP – as shown in lines 5.-6. every bounding/approximating polygon of a given 2D cluster is inserted in the vector VP. The reason we maintain both structures is as follows. If the local principal detects an occurrence of the event E_{Si}, then it can be readily reported – and if its bounding polygon is needed, the corresponding entry at $VP[i]$ can be used.

An important observation is that for some collections of

Algorithm 1 Initial Shape Detection

Input:(γ, A, t); $SN'(\subseteq SN) = \{sn_1, sn_2, \ldots, sn_m\}$ – nodes with readings above γ; proximity threshold ε
Output: List $ES = \{E_{S1}, \ldots, E_{Sk}\}$, of polygons satisfying the predicate.

1: set ε as sensingRange for DBScan and density as 1;
2: run DBScan;
3: insert each cluster in vector VC;
4: **for** each cluster of points $C_i \in VC$ **do**
5: Generate the χ-shape approximation polygon P_i;
6: Add the polygon P_i to the vector VP;
7: **end for**
8: **for** each polygon $P_i \in P$ **do**
9: calculate the area $|P_i|$
10: **if** $|P_i| \geq A$ **then**
11: insert E_{Si} into $ES(i)$
12: **else**
13: insert $NULL$ into $ES(i)$
14: **end if**
15: **end for**
16: Return ES

nodes (clusters), it may be the case that the corresponding bounding polygon for that particular cluster may fail the test in line 10. of Algorithm 1. This will generate a *NULL* entry in the corresponding position of the *ES* vector – however, we still retain the entry in the *VP* vector. The reason for it is that there may be a specific "scenario" that needs to be accounted for. Namely, it could be the case that distance between the bounding polygon P_j (such that $|P_j| < A$) and the boundary of the zone whose principal detected P_j is $\leq \varepsilon$. In such cases, there may still be an option of detecting a large-enough area warranting a detection of $E_S(\gamma, A, t)$, except a portion of it is detected by the principal of a neighboring zone.

The tests and detections of such areas spanning on each side of the boundary of the neighboring regions is performed by the parent-node of the two principals in the corresponding hierarchy. We note that in case the polygon bounding the locations of the sensors with readings above γ, but having insufficiently large areas, are located at the vicinity of the corner of the region of a particular principal, the request for detection of an instance of $E_S(...)$ may need to be propagated two levels up the hierarchy, i.e., to the grand-parent of the principal. Regardless of the level, the parent or the grand-parent will have to merge the two sets and construct their joint χ-shape approximation. In some degenerate cases – like, for example, when sensors with high-enough readings are in narrow strip along the main separating line (for K-d trees), the propagation of requests may have to continue all the way up to the root of the principals' hierarchy.

Before moving on with the dynamic case of monitoring a particular shape, we present a couple of additional observations related to Algorithm 1:

1. We note that if other parameters are added – like, for example, a maximum radius of a point-set from a given cluster, slight adjustments will need to be made in Algorithm 1. Namely, before inserting E_{Si} into $ES(i)$ (cf. line 11) we need to add an additional test and check whether the diameter of that point-set is $\leq r$.

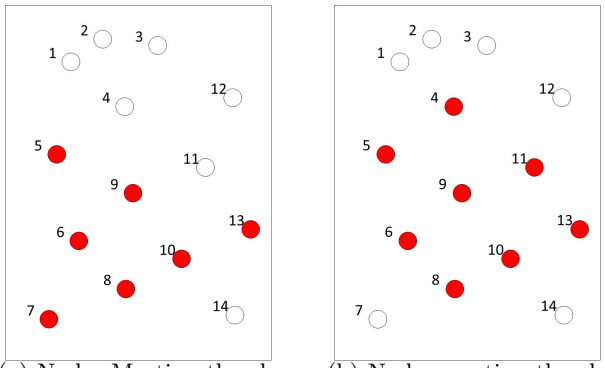
2. In some (application) scenarios it may be sufficient to use simpler representations of the point-set. For example, one may use a minimum bounding rectangle – either axes-parallel or obtained via rotating callipers algorithm [27]; or a (smallest enclosing) circle [22]. Should this be the case, the argument signature of the predicate(s) (cf. Section 2.2) can be extended by including an explicit `boundary-type`. Clearly, in such setting, instead of specifying the bound on the diameter of the point-set, one may wish to provide an alternative quality-constraint like, e.g., θ – the fraction of points inside the smallest enclosing circle with actual readings above γ.

3.2 Incremental Continuous Detection

Since the values of the phenomena at different locations may change over time, the shapes may be "mobile" (i.e., evolving). Thus, in reality, the predicate $E_S(...)$ is likely to be continuous and need re-evaluations. Acquiring messages from all the participating nodes in a periodic manner incurs communication overheads which are not necessary – and, in a similar spirit, recalculating clusters in every sampling interval may incur computations overhead. Considering the fact that WSN nodes have limited energy resources, we propose two adjustment which enable incremental re-evaluation of the continuous version of the predicate: *toggle* mode and use of an *incremental* variant of the DBScan [10].

Toggling

Although shape detection process is continuous, it is performed discretely during every sampling interval, called epoch. We observe that our sensor behave in a "binary manner" regarding the threshold value of the predicates: they either send a message when they meet the threshold or they do not send any messages. If it is the case that phenomenon is morphing at a slow rate, then it is expensive (and not necessary) for a node to send `threshold met` message for every epoch. Hence, in order to reduce communication overhead, we introduce *toggle* that makes nodes send message when the condition of their sensed value against the threshold changes. To state it differently, if a node meets a threshold for continuous epochs, it will send a message only in the first epoch during which it meets the threshold, and it will be assumed to meet the threshold in the subsequent epochs. However, when sensed value no longer meets the threshold, a message will be sent indicating to the principal node that the (location of the) particular sensor should no longer be considered when detecting the shape.



(a) Nodes Meeting threshold at t_1 (b) Nodes meeting threshold at t_2

Figure 5: Consecutive Epochs

An illustrating scenario is shown in Figure 5. Only nodes 4, 7 and 11 need to transmit a message at t_2 ($t_1 < t_2$), and all other nodes which have the same stance regarding the threshold, need not transmit anything. Thus, a total of 3 messages are sent instead of 8 messages during the transition from epoch at t_1 to epoch at t_2.

Incremental Shape Construction

Dynamic (re)configuration of clusters is a problem which has been addressed before, and there are many algorithmic solutions. As identified in [10], the impact of inserting a new point to an existing set of clusters has 4 possible outcomes:

- Noise: New point is not spatially connected to any cluster and can not form a cluster by itself, therefore considered outliers/noise.

- Creation: New point is spatially connected to a set of points, which are not part of any cluster, and form a new cluster among themselves.
- Absorption: Point is added to an existing cluster.
- Merge: New point is causes two disjoint clusters to be merged into one cluster.

On the other hand, removing a point from the dataset – which, in our case, corresponds to having a sensor with past readings exceeding γ now dropping its readings to $< \gamma$ – has 3 possible outcomes:

- Removal: Point is just removed from the point set without necessitating further processing.
- Reduction: Removed point causes a cluster to shrink
- Split: Removing the point may break the connectivity of other points, hence causing a cluster to split into two.

Now the challenge becomes how to address these insert/remove point actions in our settings. For this purpose, we use the cluster update algorithm, formalizes in Algorithm 2).

Algorithm 2 takes point P, set of all clusters, set of all points that are not part of any clusters, type of the action (whether insertion or removal), and proximity threshold ε as parameters, and outputs the updated cluster set, and set of non-clustered points. If the action is an insertion, the algorithm finds which cluster(s) this point is spatially close with ε parameter. There are 3 different procedures at this point:

1. If there are multiple clusters that this point P can be connected to, then it means this is a merge since P spatially connects two clusters, as we set density as 1. Also, it may connect other spatially close points, not previously belonging to any cluster, to this merged big cluster.
2. If there is a single cluster that this point P can be connected to, then P is simply appended to the cluster along with all non-clustered points that are spatially connected to P
3. If P is not close enough to any of the clusters, then it either forms a new cluster with other non-clustered points, otherwise P is just added to non-clustered point set.

If the action is a deletion, then algorithm finds which cluster P belongs to, and again 3 different procedures take place:

1. If P does not belong to any cluster, then it means it is in the set of non-clustered points. Therefore it is just removed from that set.
2. If P is connected with a single point, then P is just removed from the cluster.

Algorithm 2 Cluster Update Algorithm

Input: VC – vector of existing clusters; $SN''(\subseteq SN') = \{sn_1, sn_2, \ldots, sn_m\}$ – non-clustered nodes with readings above γ; Point P; *action*; proximity threshold ε
Output: VC; SN''

```
1: if action = 'insertion' then
2:   let VC_sub = indexOfConnectedClusters(VC, P);
3:   if sizeOf(VC_sub) > 1 then
4:     merge clusters in VC with indexes in VC_sub;
5:     let SN_sub = connectedPoints(SN'', P);
6:     if sizeOf(SN_sub) > 0 then
7:       add SN_sub to merged cluster;
8:     end if
9:   else
10:    if sizeOf(VC_sub) = 1 then
11:      add P to cluster VC[VC_sub]
12:      let SN_sub = connectedPoints(SN'', P);
13:      if sizeOf(SN_sub) > 0 then
14:        add SN_sub to VC[VC_sub];
15:      end if
16:    end if
17:  else
18:    if sizeOf(VC_sub) = 0 then
19:      let SN_sub = connectedPoints(SN'', P);
20:      if sizeOf(SN_sub) > 2 then
21:        create new cluster C with SN_sub and P;
22:        add C to VC;
23:      end if
24:    end if
25:  end if
26: end if
27: if action = 'removal' then
28:   let VC_i = findCluster(VC, P);
29:   if VC_i is then
30:     remove P from SN'';
31:     Return VC, SN'';
32:   end if
33:   let C_sub = connectedPoints(VC_i, P);
34:   if sizeOf(C_sub) = 1 then
35:     remove P from VC_i
36:     Return VC, SN'';
37:   end if
38:   if distance between any two points of C_sub < ε then
39:     remove P from VC_i;
40:     Return VC, SN'';
41:   else
42:     remove P from VC_i
43:     split VC_i by re-clustering;
44:   end if
45: end if
46: Return VC, SN'';
```

3. If P is connected with multiple points and if any two of those points are spatially connected, then just remove P from the cluster, otherwise split the cluster by running DBScan on this cluster.

Considering the scenario shown in Figure 5 as an example, Cluster Update Algorithm will be called three times: insertion for nodes 4 and 11, deletion for node 7. As can be seen, transition from Figure 5(a) to Figure 5(b) requires

an update on the outline of the shape as well. Therefore, after updating clusters, the χ-shape polygonal approximation is re-calculated for the ones that have changed, and the corresponding entries in the vectors EV and VP (cf. Algorithm 1) are updated.

To elaborate on the functions in Algorithm 2: (1) *findCluster* function takes a point and the vector of clusters, then returns the cluster which has the input point as a member; (2) *connectedPoints* function takes a point and non-clustered point set as arguments and returns the set of points which are within ϵ distance the the input point; (3) Lastly, *indexOfConnectedClusters* function takes a point and the vector of clusters as inputs, then returns the index of clusters, for which the input point is ϵ distant from a member of the cluster, on the vector. Our implementation of all three functions performs exhaustive distance measurement on input point and the list of points. The respective complexity of each of these functions is $O(n)$. We note that these functions are, in a sense, range queries and optimization of their processing – while desirable – is beyond the scope of this work.

We conclude this section with an observation that the boundary scenarios are equally important in the context of tracking the evolution of the shapes as they were for the initial detection (cf. Section 3.1.). It may very well be the case that the detection of an expansion of a shape (along with merging of shapes) may need to be handled by propagating corresponding requests up the hierarchy. However, as our experiments demonstrate, this still would not incur as much communication overheads as the "centralized" approach, which consists of sending individual readings to the sink.

4. DETECTING CO-OCCURRENCE OF SHAPES

We now turn our attention to detecting a co-occurrence of shapes which satisfy the area and phenomenon-value threshold, and are within certain spatio-temporal proximity. Given the definition of predicate $E_{CO}(E_{S1}, E_{S2}, d, \delta)$ (cf. Section 2), we proceed with describing the algorithmic aspects in our WSN settings. We note that the detection(s) of $E_S(\gamma, A, t)$ are assumed to have been initiated in the local principal(s) (or involving the hierarchy for boundary-conditions) before initializing the algorithms for detecting the co-occurrence predicates.

We assume that the request for detecting $E_{CO}(E_{S1}, E_{S2}, d, \delta)$ is transmitted top-down from the sink towards the lowest levels of the hierarchical index – down to the local principals. Similarly to the detection of $E_S(\gamma, A, t)$, we distinguish between initial detection and "tracking" tracking of the co-occurrence predicate in a continuous manner.

The initial evaluation of co-occurrence predicates follows the shape detection predicate detection. Having detected shapes and put into a vector, principal traverses through its list of co-occurrence predicate queries. For each query, polygons of satisfying shape predicates are extracted and Euclidean distance between these polygons is calculated via edge to edge, vertex to vertex, edge to vertex distance computations. If distance between the polygons is smaller than E_{CO}'s d parameter and occurrence of shapes are within the time interval δ, then E_{CO} predicate holds and it is reported in the hierarchy.

Algorithm 3 Co-occurrence Detection

Input: List EV; list of predicate occurrences of type E_S, List VC_q; list of predicates $E_{CO_i}(E_{S1_i},\ E_{S2_i},\ d_i,\ \delta_i)$ requested

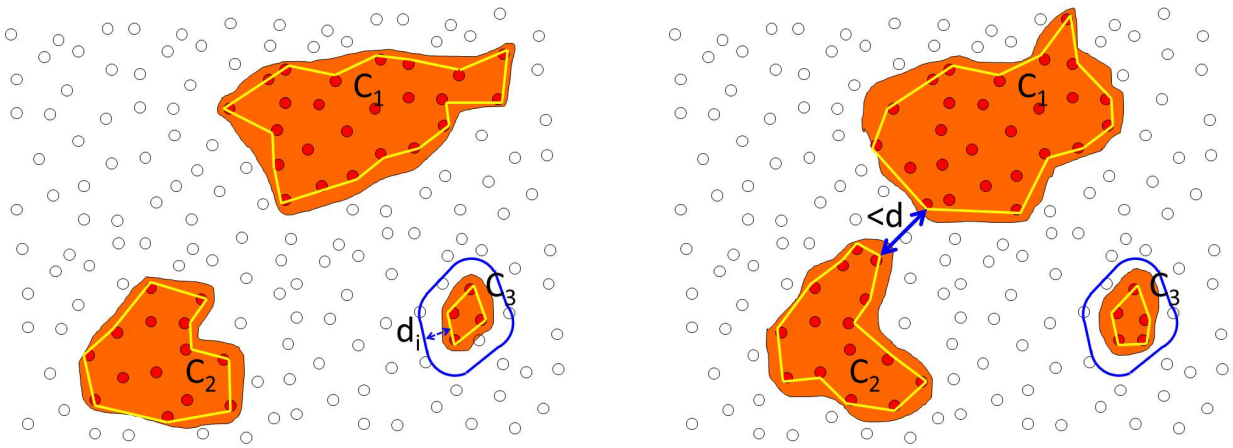
Output: List VC_o of predicate occurrences of type $E_{CO}(...)$

1: **for** all composite predicate EC_{CO_i} queries in VC_q **do**
2: **for** every pair E_i and E_j in VE **do**
3: compute the Euclidean distance between closest points of polygons
4: **if** distance$(E_i,\ E_j) < d$ AND $|E_i.t - E_j.t| < \delta$ **then**
5: insert the occurrence of detected EC_{CO_i} in VC_o
6: **end if**
7: **end for**
8: **end for**
9: Return VC_o

If we take the exemplary scenario in Figure 6(a), there are three spatial regions of interest in the principal's jurisdiction. The nodes highlighted by red disks will be the ones sending the message to the principal. Subsequently, principal clusters the group of points (denoted C_1, C_2, C_3) based on their proximity, and calculates the polygon for each cluster by running Algorithm 1 with reaching to conclusion that each polygon satisfy a specific shape predicate E_{S_i}. However, shapes are too far from each other to satisfy the distance parameter for any $E_{CO}(...)$ predicate. As the time flows, shapes evolve for the next epoch and principal's point of view becomes as in Figure 6(b). Cluster Repair Algorithm (Algorithm 2) is run to update the clusters C_1, C_2 and C_3, at which point the bounding polygons are recalculated. Afterwards, co-occurrence predicate detection process is re-initiated, and minimum Euclidean distance between polygons representing clusters C_1 and C_2 turns out to be smaller than d parameter of a queried co-occurrence predicate $E_{CO_i}(E_{S_i}, E_{S_i}, d, 0)$. In this case, $E_{CO_i}(...)$ predicate is determined to hold and sink is notified. One thing to note is that E_{CO} predicates, by default, are executed for simultaneous occurrence, meaning $\delta = 0$, and experiments were done under this circumstance. However, historic predicate occurrence data needs to be stored for different δ values.

Evaluation of co-occurrence predicate is formalized in Algorithm 3. The input to the algorithm executed by the local principals consists of the list of the $E_S(...)$ events currently valid and the list of requests of $E_{CO}(...)$ events to be detected. We note that the list of $E_S(...)$ is already available from the vector ES generated by Algorithm 1. Once again, we used vectors to implement the corresponding lists.

Similarly to Algorithm 1 it may be the case that some shapes satisfying $E_S(\gamma, A, t)$ could contribute to co-occurrence detection with shapes belonging to regions managed by other principals. In such cases, the request for detecting $E_{CO}(...)$ will have to be propagated up the hierarchy of principals. In the worst case, the propagation may need to be recursively carried all the way up to the root – however, to speed up



(a) Shapes at t_1

(b) Shapes at t_2

Figure 6: Detecting contiguous evolving shapes

the computation we propose the following pruning heuristics to be used by the inner nodes/principals: if the distance between the closest polygons in the children is larger than the largest d_i from the set of pending requests for detecting $E_{CO}(...)$, then the parent node need not propagate the request further up the hierarchy, nor does it need to apply the corresponding instance of Algorithm 3.

As the phenomena values fluctuate, the size and location of the individual shapes will evolve over time. This, in turn, may affect the co-occurrence predicates: some of which were valid may become invalid and vice-versa – some instances which did not hold, may become true. To avoid repeated invocations of Algorithm 3 with every sensing epoch, we propose the following event-based management of the continuous variant of $E_{CO}(...)$:

1. Expand the convex hull of the polygonal boundary P_i of every $E_{Si} \in VE$ by d_j, where $d_j/2$ is the smallest of all the distances in the current list of pending $E_{CO}(...)$'s. In other words, calculate the Minkowski sum $Hull(P_i) \oplus d_j/2$.

2. If the evolution of P_i is within $P_i \oplus d_j/2$, then there is no need to re-evaluate any $E_{CO}(...)$ predicate with respect to P_i at the new time-instant, for any other shape E_{Sk} for which $(Hull(P_k) \oplus d_j/2) \cap Hull(P_i) \oplus d_j/2 = \emptyset$. The reason is that if P_i and P_j were not close enough before and neither of them has evolved outside the "safety" region – they are guaranteed not be close-enough after the updates.

For instance, blue outline in Figure 6(a) represent the Minkowski sum $Hull(P_3) \oplus d_j/2$ at $t = t_1$, where P_3 represents the polygon for the cluster C_3. Observing the fact that P_3 is within the blue outline at $t = t_2$, it is pruned for further executions of co-occurrence detection process. As the experimental observations will demonstrate, the pruning along the hierarchy and the use of "safe" regions improve the communication overheads in our proposed approaches.

5. EXPERIMENTAL EVALUATION

Our experimental evaluations were run using our open-source WSN simulator, SIDnet-Swans [14]. We considered a WSN consisting of 800 homogeneous nodes, each being capable to sense the value of a simulated phenomenon at its location. The deployment of the sensors was semi-random in the sense that we controlled the value of the discrepancy of their distribution within the cells. All experiments were conducted at a field with $1000 \times 1000m^2$, with the radio transmission strength 12 dBm, and the sampling frequency 5 seconds.

We used synthetic generation of the temperature phenomenon for our experiments, whereby the region of interest was split into an 8×8 grid. With a pre-defined frequency, a new matrix is being created with every grid assigned random values between 0 and 100 degrees Celcius, and the current matrix changes its values to this newly created target matrix linearly over time. Increasing the frequency of the new matrix creation yields higher fluctuations of the values of the phenomenon, where decreasing will make it more stable. We note that although the geographic field is split into 64 cells, the values within a particular cell are not uniformly same – rather, they are calculated using bi-linear interpolation

Our experimental results are the average of 3 runs with same setups. We ran shape detection predicates with different γ values. Phenomenon creating and evolution is random and can vary for different runs, except for the last part of the experiments where all experiments were run using the same phenomenon.

We report the comparison between our proposed methodologies; centralized contour tracking algorithm; and distributed periodic contour tracking algorithms, as these methods are used for comparison in previous works [33]. Centralized contour tracking algorithm simply uses the sensed value the phenomenon in individual sensors and each of those values are transmitted to the sink via shortest geographic path routing scheme (only the values satisfying the preset threshold). Distributed periodic contour tracking is constructed

on the hierarchical routing scheme and periodically reconstructs the contour with new set of data sent every epoch. We note that we measure communication expenditure by the total number of message-hops exchanged between nodes, as data is transmitted in a multi-hop manner in the network.

First set of observations aims to reveal that our distributed approach saves substantial communication overhead when compared to centralized approach in Figure 7. Figure 7 shows number of message hops over time for two different approaches. Note that *toggle technique* is not applied in either type of the experiments. Same set of predicates were applied for both experimental setups with the signature $(93, 300m^2, t = now)$. Our findings show that by distributing the predicate detection among local principals, 4 times less communication overhead is incurred, when compared to the centralized approach (*CENT* refers to centralized approach, *DIST* refers to distributed approach).

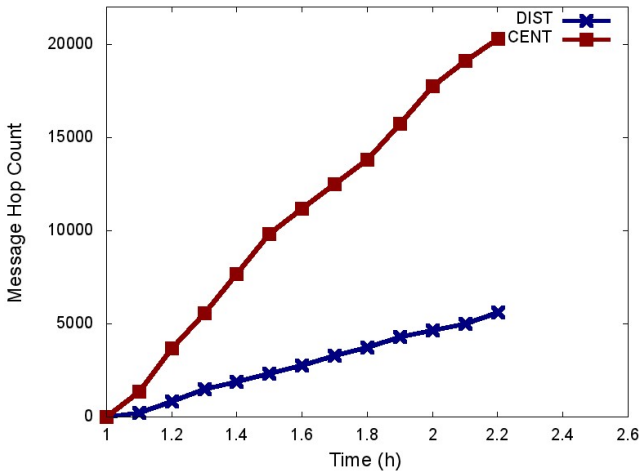


Figure 7: Centralized vs Distributed

Our next group of experiments aims at illustrating benefits of *toggle technique*. In this setups, we used distributed approach and we implemented *toggle technique* in one of them and did not implement on the other to observe the benefit of the incremental updates. Experiments were run with 3 different instances of the predicates, corresponding to 3 different γ values: 80°C, 85°C, 90°C. Figure 8 shows the number of message hops for each experimental setups. Even though total number of messages vary, the relative benefits of toggling are clear – from 2 times up to 3 times smaller communication overhead.

In addition, Figure 9 illustrates the effect of *toggle* technique under the effect of fluctuation rate of the phenomenon. As discussed earlier, toggle mechanism saves more communication overhead if phenomenon fluctuation rate is low when compared to regular periodic transmission approach. Figure 9 shows the communication overhead produced by regular approach divided by communication overhead incurred after applying our toggle mechanism approach. $y-axis$ represent the new phenomenon creation frequency. For example, with 2 hours setting, a new phenomenon will be created every 2 hours and the current phenomenon will morph into the new phenomenon linearly.

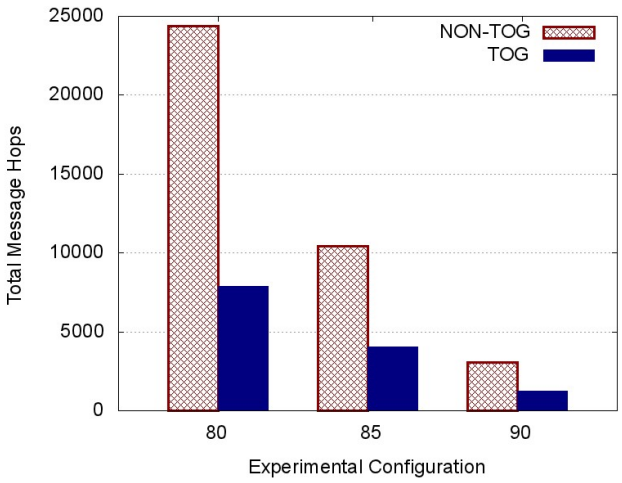


Figure 8: Toggle Mechanism Benefits

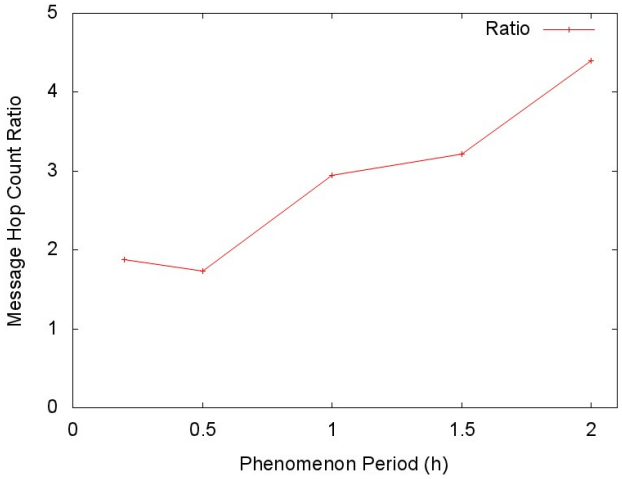


Figure 9: Fluctuation Ratio Effects

Our last set of experimental results aims at quantifying the benefits of co-occurrence detection when distributing the work among local principals, as opposed to applying centralized detection in the sink (root of the hierarchy). Figure 10 shows the communication cost comparison between centralized approach and distributed approach while all other techniques are being applied. We note that the same phenomenon values were used in all experiments to ensure fairness. In both approaches, the individual predicates' detection is performed in the same way and their communication overheads are not added to the total cost. Centralized approach is similar to the one in [33], where boundaries are detected in a distributed manner, and boundary information has to be sent to the sink for co-occurrence. The predicate used in this experimental settings has different γ values: 80, 85, 90, area is set to 300 square meters with signatures $E_S(80, 300m^2, t = now)$. Co-occurrence predicate was set to detect any two predicate occurrences happening at the same time with signature $(E_S, E_S, 100, 0)$. *CENT* refers to centralized co-occurrence detection scheme and *DIST* refers to approach where distributed co-occurrence detection is per-

formed among local principals.

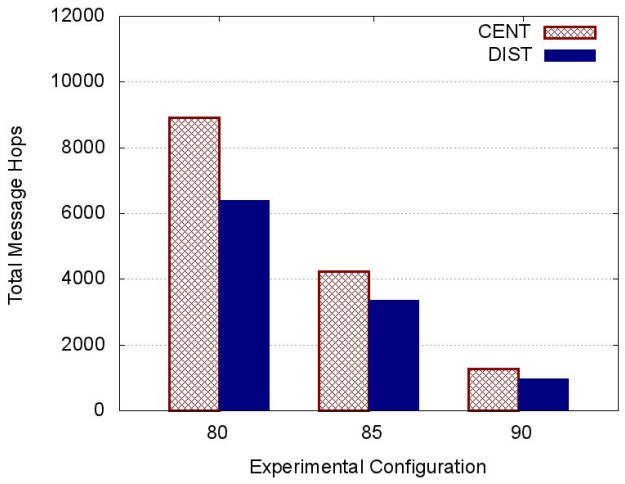


Figure 10: Centralized vs. Distributed Co-occurrence

6. RELATED WORK

There are several bodies of research results which are related to our work.

Topological methods for boundary (and holes) detection are presented in [29] – however, the work focuses on static scenario, in the sense that the values read by particular sensors are not formally considered. The problem of tracking dynamic boundaries has been addressed in several works. In [13], spatial summary of the phenomenon is acquired in a distributed manner an isocontour map of the field is created. Other techniques for estimating the boundaries of a dynamic region are proposed in [7]. In a similar spirit [33] introduces a light-weight contour tracking method for binary binary sensors, and proposes methodologies for merging and splitting actions over the contours. The methodologies are completely distributed, having in mind the minimization of the communication overhead. Complementary to these works, we introduced spatial predicates which pertain to threshold-based detection (and approximation) of shapes and their continuous monitoring. In addition, we investigated a distance-based relationship of distinct such shapes and demonstrated that adding a hierarchical structure may help in reducing the communication overhead when detecting the predicates.

The DRAGON protocol introduced in [17] pursues a similar objective to ours, but from a complementary perspective. The work proposes the concept of a center of mass as a representative of regions in which certain phenomena (readings of sensors) is detected, and addresses the problems of merging and splitting such shapes and generating new centers. In our work, we have focused on detecting predicates which are based on the area (size) of the region in which a particular phenomenon exceeds a given threshold and proposed methodologies for event-based validation. A qualitative description and a method for detecting homogeneous shapes in WSN settings was presented in [20]. In our work, we used χ-shapes to derive the bounding polygons of what corresponds to homogeneous regions in [20] and we also proposed a hierarchical solution for detecting a distance-based relationships between pairs of regions.

There is a substantial body of works addressing the co-location and co-occurrence problems in data mining and pattern recognition communities. Spatio-temporal co-occurrence pattern mining was addressed in [4], and there are more recent works[24, 25] attempting to solve spatio-temporal co-location pattern discovery. Interestingly, [24] addressed a specific setting of two dimensional shapes representing solar data and it bears similarity to our objective. However, most of data mining and pattern recognition works focus on historic data, and they do not cope with the evolution of the approximation-shapes.

7. CONCLUSION AND FUTURE WORK

We presented a methodology for tracking evolving spatial shapes in WSNs, where threshold-based criteria for sensed values was used to define a particular shape. We formalized the concept of a shape in terms of the locations of the participating sensors and presented efficient algorithms for initial detection and tracking. In addition, we also presented efficient solution to the problem of detecting a co-occurrence of evolving spatial shapes. Our techniques were based on a two-phase approach where part of the detection process was done locally, within a region governed by a selected principal – and, when needed, the detection process would proceed along the hierarchy of such principals. We presented experimental evaluations which demonstrated that our proposed approaches outperform the naïve one – where all the individual readings are transmitted to a dedicated sink – in terms of communication/energy savings.

There are two main directions of our future work. Part of our investigation is geared towards the "completeness" aspect of our techniques, in the sense of providing efficient algorithms for managing a complete suite of predicates such as *overlap*, *contain*, etc... (cf. [9]) for evolving spatial shapes. One of the main challenges here is to properly incorporate the values of the threshold γ in the semantics and investigate different subsumption properties. Namely, a non-empty intersection of two shapes with 70C and 100C thresholds is an indication for merging their polygons w.r.t. 70C, but not w.r.g. 100C. A complementary avenue that we plan to pursue is investigation of a context-aware speed up techniques. One immediate step is to provide a sorting of the predicates based, e.g., on Z-curves, for the purpose of more efficient pruning (cf. Section 4.). Another aspect of the efficiency is the selection of indexing structures. More specifically, we believe that properties such as non-uniform distribution of the nodes, incorporating mobile nodes, and dynamic adjustment of Quality of Service (i.e., increased coverage or tracking quality) may require more flexible/adaptable indexing techniques.

8. REFERENCES

[1] R. Adaikkalavan and S. Chakravarthy. Event specification and processing for advanced applications: Generalization and formalization. In *Database and Expert Systems Applications*, pages 369–379, 2007.

[2] K. Akkaya and M. Younis. A survey on routing protocols for wireless sensor networks. *Ad Hoc Networks*, 3(3), 2005.

[3] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: A survey. *Comput. Netw.*, 38(4):393–422, Mar. 2002.

[4] M. Celik, S. Shekhar, J. P. Rogers, and J. A. Shine. Mixed-drove spatiotemporal co-occurrence pattern mining. *IEEE Trans. on Knowl. and Data Eng.*, 20(10):1322–1335, Oct. 2008.

[5] I. Demirkol, C. Ersoy, and F. Alagoz. Mac protocols for wireless sensor networks: A survey. *IEEE Communication Magazine*, 2006.

[6] M. Duckham, L. Kulik, M. Worboys, and A. Galton. Efficient generation of simple polygons for characterizing the shape of a set of points in the plane. *Pattern Recogn.*, 41(10):3224–3236, Oct. 2008.

[7] S. Duttagupta, K. Ramamritham, and P. Kulkarni. Tracking dynamic boundaries using sensor network. *Parallel and Distributed Systems, IEEE Transactions on*, 22(10):1766–1774, Oct 2011.

[8] H. Edelsbrunner, D. Kirkpatrick, and R. Seidel. On the shape of a set of points in the plane. *Information Theory, IEEE Transactions on*, 29(4):551–559, Jul 1983.

[9] M. Erwig and M. Schneider. Spatio-temporal predicates. *IEEE Trans. on Knowl. and Data Eng.*, 14(4):881–901, July 2002.

[10] M. Ester, H.-P. Kriegel, J. Sander, M. Wimmer, and X. Xu. Incremental clustering for mining in a data warehousing environment. In *PROC. 24TH INT. CONF. VERY LARGE DATA BASES, VLDB*, pages 323–333, 1998.

[11] M. Ester, H. peter Kriegel, J. S, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. pages 226–231. AAAI Press, 1996.

[12] M. Fayed and H. T. Mouftah. Localised alpha-shape computations for boundary recognition in sensor networks. *Ad Hoc Netw.*, 7(6):1259–1269, Aug. 2009.

[13] S. Gandhi, J. Hershberger, and S. Suri. Approximate isocontours and spatial summaries for sensor networks. In *Proceedings of the 6th International Conference on Information Processing in Sensor Networks*, IPSN '07, pages 400–409, New York, NY, USA, 2007. ACM.

[14] O. C. Ghica, G. Trajcevski, P. Scheuermann, Z. Bischof, and N. Valtchanov. Sidnet-swans: A simulator and integrated development platform for sensor networks applications. In *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*, SenSys '08, pages 385–386, New York, NY, USA, 2008. ACM.

[15] C. Hartung, R. Han, C. Seielstad, and S. Holbrook. Firewxnet: a multi-tiered portable wireless system for monitoring weather conditions in wildland fire environments. In *MobiSys*, 2006.

[16] A. Hinze and A. Voisard. A parametrized algebra for event notification services. In *TIME*, 2002.

[17] N. Hubbell and Q. Han. Dragon: Detection and tracking of dynamic amorphous events in wireless sensor networks. *Parallel and Distributed Systems, IEEE Transactions on*, 23(7):1193–1204, July 2012.

[18] K. Iwanicki and M. van Steen. A case for hierarchical routing in low-power wireless embedded networks. *TOSN*, 8(3):25, 2012.

[19] J. Jeong, T. Hwang, T. He, and D. H.-C. Du. Mcta: Target tracking algorithm based on minimal contour in wireless sensor networks. In *INFOCOM*, 2007.

[20] J. Jiang, M. Worboys, and S. Nittel. Qualitative change detection using sensor networks based on connectivity information. *GeoInformatica*, 15(2):305–328, 2011.

[21] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. Tinydb: An acquisitional query processing system for sensor networks. *ACM Trans. Database Syst.*, 30(1):122–173, Mar. 2005.

[22] N. Megiddo, N. Megiddo, N. Megiddo, and N. Megiddo. Linear-time algorithms for linear programming in r3 and related problems. In *Foundations of Computer Science, 1982. SFCS '08. 23rd Annual Symposium on*, pages 329–338, Nov 1982.

[23] M. M. A. Mohamed, A. A. Khokhar, and G. Trajcevski. Energy efficient in-network data indexing for mobile wireless sensor networks. In *SSTD*, pages 165–182, 2013.

[24] K. G. Pillai, R. A. Angryk, and B. Aydin. A filter-and-refine approach to mine spatiotemporal co-occurrences. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL'13, pages 104–113, New York, NY, USA, 2013. ACM.

[25] F. Qian, Q. He, K. Chiew, and J. He. Spatial co-location pattern discovery without thresholds. *Knowledge and Information Systems*, 33(2):419–445, 2012.

[26] R. Szewczyk, A. M. Mainwaring, J. Polastre, J. Anderson, and D. E. Culler. An analysis of a large scale habitat monitoring application. In *SenSys*, pages 214–226, 2004.

[27] G. Toussaint. Solving geometric problems with the rotating calipers. *In Proc. IEEE MELECON âĂŹ83*, pages 10—02, 1983.

[28] M. Umer, L. Kulik, and E. Tanin. Spatial interpolation in wireless sensor networks: localized algorithms for variogram modeling and kriging. *GeoInformatica*, 14(1):101–134, 2010.

[29] Y. Wang, J. Gao, and J. S. B. Mitchell. Boundary recognition in sensor networks by topological methods. In *MOBICOM*, pages 122–133, 2006.

[30] G. Werner-Allen, K. Lorincz, M. Welsh, O. Marcillo, J. Johnson, M. Ruiz, and J. Lees. Deploying a wireless sensor network on an active volcano. *IEEE Internet Computing*, 10(2), 2006.

[31] A. Woo, S. Madden, and R. Govindan. Networking support for query processing in sensor networks. *Communications of the ACM*, 47:2004, 2004.

[32] F. Zhou, G. Trajcevski, O. Ghica, R. Tamassia, A. Khokhar, and P. Scheuermann. Deflection aware tracking principal selection in active wireless sensor networks. *IEEE Trans. on Vehicular Technology*, 61(7), 2012.

[33] X. Zhu, R. Sarkar, J. Gao, and J. Mitchell. Light-weight contour tracking in wireless sensor networks. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pages –, April 2008.