2010 16th International Conference on Parallel and Distributed Systems

Distributed Spatial Analysis in Wireless Sensor Networks

Farhana Jabeen and Alvaro A. A. Fernandes
School of Computer Science
University of Manchester
Manchester M13 9PL, UK
Email: {f.jabeen,a.fernandes}@cs.man.ac.uk

Abstract— **Environmental monitoring is an important application area for wireless sensor networks (WSNs). An important problem for environmental WSNs is the characterization of the dynamic behaviour of transient physical phenomena over space. In the case of mote-level WSNs, a solution that is computed inside the WSN is essential for energy efficiency. In this context, the main contributions of this paper to the literature on in-network processing in WSNs are threefold. The paper further develops an algebraic framework with which one can express and evaluate complex topological relationships over geometrical representations of permanent features (e.g., buildings, or geographical features such as lakes and rivers) and of transient phenomena (e.g., areas of mist over a cultivated field). The paper then describes distributed implementations of spatial-algebraic operations over the regions represented by that framework, thereby enabling identification of topological relationships between regions. Finally, the paper presents experimental evidence that the techniques described lead to efficient runtime behaviour. Taken together, these contributions constitute a further step towards enabling the high-level specification of expressive spatial analyses for efficient execution inside a WSN.**

I. Introduction

There is widespread agreement that WSNs are on the way to becoming an essential technology for monitoring the natural environment and for modelling the dynamic behaviour of transient physical phenomena over space, e.g., close to fifty environmental sensor networks were surveyed in [8]. A WSN may also be viewed as a distributed computing platform [2] if one construes each node as a networked computational resource, albeit a very constrained one. It is an integral part of the latter view that nodes in a WSN cooperate in the execution of high-level tasks, e.g., query processing and data analysis. In this context, the main goal of the paper is to contribute to the enabling of in-network distributed spatial analyses of transient spatial phenomena (e.g., a shape-shifting region of mist) that can be sensed using WSNs. In particular, the paper focuses on the challenges involved in using WSNs to identify, track and report topological relationships between dynamic, transient spatial phenomena and asserted regions (e.g., whether the mist is adjacent, or inside, or outside a cultivated field) in the specific case where the regions involved can be characterized by sets of nodes.

The motivation for the research stems primarily from applications where important decisions hinge on the detection of an physical phenomena, e.g., in precision agriculture. Precision agriculture is a developing discipline aiming to enhance farming efficiency [13]. One real-world example in which WSNs are being deployed for precision viticulture is at Camalie Vineyards [22]. Monitoring the physical quantities such as moisture in the soil, and temperature is crucial to improving crop yield and quality. The availability of real-time information makes it possible to take managerial decisions promptly such as to target irrigation when soil in one area dries up, or to change the schedule for using pesticides. Instead of retrieving raw data for farmers, we aim to make the WSN respond to the information requests by the farmers that require exploring the relationships between spatially-referenced entities, and to derive representations grounded on such relationships. As the basis for our examples and experiments, we have used the underlying regions (see Fig. 1) in this particular deployment[1]. For example, given thresholds θ and θ', call `M` the event region where soil moisture is above θ and `T` the event region where temperature is below θ'. Fig. 1, shows asserted regions (i.e., those that represent stable features of the physical world, like fields) ($f1$-$f10$) and event regions `M` and `T`. Our motivation is to enable WSNs that allow a user to pose a task (expressed as a spatial-algebraic expression) to retrieve the required information. One such could determine whether there is a need to spray a field, say `f5`. This is a spatial-analytic task that can be expressed in terms of the `intersects` topological relationship as follows:

```
((M Intersects f5) AND
   ((T Intersects f5) AND (M Intersects T)))
```

If this expression returns **true**, the farmer needs to take action. In order to derive the precise region where spray is needed, the following task can be posed:

```
((M Intersection f5) Intersection (M Intersection T))
```

Note that while the first task above is Boolean-valued expression, the second returns a derived geometry, viz., the region of space where the first task yielded **true**. The operations in the tasks above fall into two groups: *spatial-valued operations* return a derived geometry (e.g., `minus`, `plus`, `intersection`), *Boolean-valued operations* characterize topological relationships (e.g., `intersects`,

[1]More information is available at `http://camalie.com/WirelessSensing/WirelessSensors.htm` [Accessed: 27 Sep 2010]. In particular, see `http://camalie.com/CamalieGIS/Naked/` [Accessed: 27 Sep 2010] for the geometries.

1521-9097/10 $26.00 © 2010 IEEE
DOI 10.1109/ICPADS.2010.58

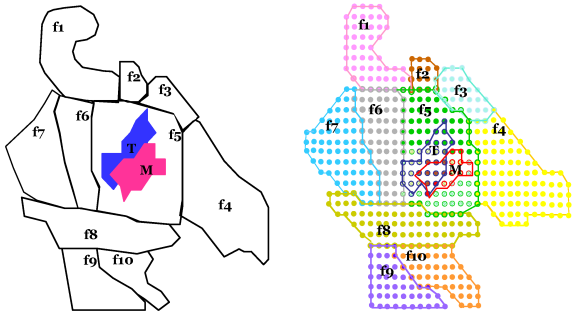

Fig. 1. *(a.)* Fields (f1-f10) in a vineyard showing approximate node-field membership *(b.)* Example WSN over (*a.*)

`area_disjoint`, `vertex_disjoint`). In this paper, we focus on Boolean-valued operations. Spatial-valued operations are the focus of [10].

The remainder of the paper is structured as follows. We discuss related work in Sec. II, which also introduces the technical background for the framework contributed in Sec. III for representing regions in WSNs. Sec. IV describes our proposed algebra for characterizing complex and expressive topological relationships over WSNs. We discuss task specification in Sec. V. Sec. VI describes how spatial analysis based on the framework and algebra can be evaluated by completely distributed computations. An empirical evaluation of the algorithms is presented in Sec. VII and we conclude in Sec. VIII.

II. Related Work

To the best of our knowledge, the representational framework, the spatial algebra over that framework and the distributed implementation of the latter described in this paper and in [10] constitute the first comprehensive, implemented, empirically evaluated proposal for in-network computation of topological relationships in WSNs. In [22], [8], each and every sensed measurement is transmitted towards a gateway. Three boundary detection methods are studied in [28]. Their evaluation approach is to send boundary information to a gateway, so that the gateway can generate the snapshot of the event region based on the received data. The work described in [24] provides a computational model for WSNs to detect topological change (i.e., hole formation, hole disappearance, event region splitting, and event region merging) in dynamic regions based on local low-level snapshots of spatio-temporal data. Jiang et al. [24] focus on using the connectivity information instead of location information for detecting topological change. Other works that use WSN to detect topological change includes [5], [11]. A prerequisite for detecting topological change is event detection and subsequent derivation of an event boundary. Jiang and Worboys [11] propose in-network algorithms for the detection and reporting of topological changes. Zhu et al. [1] provide a light-weight contour detection algorithm, to track the contours of the evolving region, to get useful information about the topological features. In [26], the authors present an algorithm for the retrieval of topology at multiple resolutions. Bi et al. [27] propose an algorithmic strategy for the detection of topological holes.

The framework proposed in [15] dynamically builds a structure over the nodes that belong to the event region for information collection and aggregation. The work reported in [23] focuses on providing a high-level programming interface that abstracts away the details of routing, data collection, data dissemination, and state management. The authors propose a region abstraction for programming WSNs defined in terms of radio connectivity, geographic location, or other properties of nodes. In contrast, our work supports much more expressive algebraic abstractions with expressive operations upon them.

Point set topology provides the most widely used mathematical tools for formalizing the topological relationships between spatial objects, the most expressive result of which is [4]. Whilst point set topology offers an elegant formalization of spatial relationships, it is difficult to implement efficiently. In response to this, Schneider and Güting developed the ROSE algebra [20] for implementation in spatial database systems (one such implementation is described in [7]). The ROSE algebra has as spatial data types points, lines, and regions on the plane. Over such types, a comprehensive set of algebraic operations is defined. Schneider and Güting provided efficient centralized algorithms for the ROSE-algebraic operations [20]. The ROSE algebra has desirable characteristics for the purposes of the work described in this paper, viz., it rigorously defines a comprehensive set of Boolean-valued and spatial-valued operations and it supports complex regions such as regions containing holes, and multi-element ones. The algebra we have characterized is closely inspired by and reuses most of the formal work done by Schneider and Güting. We envisage that our contributions are most useful in application contexts where analyses are periodically performed (e.g., using continuous, in-network query processing technologies [17], [25], [6]). However, the algorithms in [20], though efficient for centralized execution of one-off execution over stored data, are not usable for in-network processing in WSNs, where execution is distributed and carried out periodically over sensed data streams. The ensuing challenges are highly nontrivial, particularly so in the case of transient, dynamic regions. In such scenarios as described in Sec. III, a task is re-evaluated with some periodicity and each node independently updates the local information that defines the event regions they are members of. Furthermore, a node must cooperate with other nodes insofar as it does not have access to information other than that which it holds. Therefore, complete information regarding event regions is distributed throughout the WSN. This means that in-network spatial analysis is a highly-distributed problem.

III. Framework for Distributed Spatial Analysis over WSNs

In this paper, to a WSN there corresponds a finite, discrete, two-dimensional space that models a spatial domain. All our contributions, reported here and in [10], apply over points, lines and regions (extending those introduced in [20]). However, for reasons of space, henceforth the paper focusses exclusively on regions. Let $\boldsymbol{M} \times \boldsymbol{N}$ denote the two dimensional Euclidean plane that discretizes a geographic area G under

study. A set of nodes S is deployed inside G and the overall disposition of nodes may be regular (i.e., grid-like) or not. It is assumed that a node is location-aware and its position in the WSN can be determined. We assume that the sensing and communication range of a node $s_i \in S$ is representable by a circle with radius $r^d_{s_i}$ and $r^c_{s_i}$ respectively. This determines a WSN connectivity graph whose vertices are the deployed nodes and whose edges consist of pairs of nodes such that each element in the pair is capable of communicating with the other. A link is established if two nodes directly receive messages from each other. Each node has a limited communication range. This in turns allows regions to be defined over the WSN as subgraphs of the WSN connectivity graph. Note also that it may or may not be the case that there is a node in every point in the grid, but regions are always defined in terms of deployed nodes. The size and shape of the dynamic, transient phenomena varies over time and space. If a large number of nodes can be deployed, finely-grained event detection is possible. Otherwise, one has to compromise on the precise shape and size of the event region (i.e., transient phenomenon). Following the ROSE-algebraic approach [20], application-specific regions can be defined based on abstract data types such as **regions** (a spatial value of which could denote, e.g., a building, or a cultivated field). In our framework, regions are represented in terms of deployed nodes. Each sensor node is aware of every region it is a part of and stores region-related information locally in a table, called the *geometric information table* (GIT). A node can be part of one or more regions. Therefore, a node assigns a unique *Geometric ID* (GID) to each region it belongs to. Given a node s_i, each entry in the GIT is a triple $\langle GID, BN, TTL \rangle$, where BN is **true** iff s_i is on the boundary of the region, and TTL denotes the time-to-live after which the GIT entry for that region becomes invalid. The timer component allows a node to update the TTL of each entry in GIT.

Our framework considers three kinds of regions, viz., asserted, induced, and derived. *Asserted* regions are representations of physical features (e.g., a cultivated field). *Induced* regions are characterized by means of event detection and boundary computation. Briefly, by *event detection* it is meant that the outcome of a distributed process in which a node evaluates the event-defining predicate (e.g., `temperature>10`) and, as an outcome, declares itself an event node (if the predicate is satisfied) or not. The boundary of the event region can then be computed using a distributed boundary detection algorithm (see [9] for a study). The boundary detection algorithm allows the nodes to determine whether they lie on the boundary or in the interior of an induced region. *Derived* regions are obtained by applying spatial-valued operations (e.g., `plus`, `minus`) to existing regions. We refer the reader to [10] for more information on how derived geometries are handled in our approach.

Asserted regions are both assumed to pre-exist the WSN deployment and to remain unchanged for the duration of that deployment. Therefore, the TTL for asserted regions is set to ∞ and not updated. Their GID is also assigned once, at deployment time. Induced regions are assumed not to pre-exist the WSN deployment. The entry for an induced region in a GIT is maintained dynamically. When a node declares itself an event node, it updates its GIT as follows. If an entry already exists for that event region, its TTL is reset (typically, in the case of periodic evaluation, until the next evaluation period). Otherwise, a new entry is added to the GIT. Given that one (or both) of the arguments can be induced, nodes must maintain their membership status dynamically. The TTL is set to the minimum TTL of the original operands. Upon the expiry of the TTL of an induced geometry, its entry is removed from the GIT. The GIT allows a node to keep an up-to-date record of the regions it belongs to. This is used, e.g., to decide whether the node should evaluate some task at a specific evaluation period (if it does not belong to the relevant regions, it will not). The GID for induced regions is passed as a parameter of the task message, i.e., the one that injects an algebraic expression into the WSN for evaluation.

The proposed framework not only allows for the periodic detection of the induced regions in order to accompany the evolving nature of dynamic, transient phenomena: it also supports periodic evaluation of tasks that determine whether a spatial predicate holds. As in [17], in our system, the evaluation period (e.g., every 30 minutes) and the overall duration for the task are passed as parameters of the task message. Energy efficiency is achieved because information is kept inside the WSN, i.e., information about these regions is not sent back at the end of every evaluation period. Spatial tasks are evaluated periodically and only the outcome of task evaluation is sent back.

IV. Formal Definitions and Classification of Topological Relationships

A region is a (possibly empty, possibly singleton) finite set of pairwise disjoint triples of the form $\langle b, i, H \rangle$ where b is a cycle denoting the *boundary* of the region, i denotes the *interior* of the region and is a set that contains all the nodes enclosed by b excepting those belonging to H, and H is a (possibly empty) finite set of pairwise disjoint regions lying in the interior of b, each element of which denotes a hole in the region. Thus, the boundary and the exterior of a region with holes are allowed to be disconnected (see Fig. 6). Therefore, a region with holes has one outer boundary and one or more interior boundaries based upon the number of disjoint holes lying inside the region. Each inner boundary delimits one hole in the region (see Fig. 6). Note that the nodes enclosed by elements of H *do not* belong to i, and hence not to the region either. A node only keeps information about a region if it lies in the interior or in a (inner or outer) boundary of that region. A *unit* region denotes a region without interior. More formally, it is defined as a (possibly singleton) finite set of pairwise disjoint, polygon values having an empty interior. A *minimal-unit* region, is the smallest unit region. It is defined as a (possibly singleton) finite set of triangles having an empty interior. A region may comprise a single element, in which case we refer to it as a *single element region* (SER), otherwise as a *multi-element region* (MER).

Let r and r' denote SERs with or without holes or unit values, and R and R' denote MERs whose constituents are

R $Vertex\_Disjoint$ R' $\equiv \ \forall r \in R \forall r' \in R'$:r $Vertex\_Disjoint\ r'$
R $Area\_Disjoint$ R' $\equiv \ \forall r \in R \forall r' \in R'$:r $Area\_Disjoint\ r'$
R $Edge\_Disjoint$ R' $\equiv \ \forall r \in R \forall r' \in R'$:r $Edge\_Disjoint\ r'$
R $Adjacent$ R' $\equiv \ R\ Area\_Disjoint\ R' \wedge \exists r \in R \exists r' \in R'$: r $Adjacent\ r'$
R $Meets$ R' $\equiv \ R\ Area\_Disjoint\ R' \wedge \exists r \in R \exists r' \in R'$: r $Meets\ r'$
R $Area\_Inside$ R' $\equiv \ \forall r \in R \exists r' \in R'$:r $AreaInside\ r'$
R $Edge\_Inside$ R' $\equiv \ \forall r \in R \exists r' \in R'$:r $Edge\_Inside\ r'$
R $Vertex\_Inside$ R' $\equiv \ \forall r \in R \exists r' \in R'$:r $Vertex\_Inside\ r'$
R $Border\_In\_Common$ R' $\equiv \ \exists r \in R \exists r' \in R'$:r $Border\_In\_Common\ r'$
R $Intersects$ R' $\equiv \ \exists r \in R \exists r' \in R'$:r $Intersects\ r'$

TABLE I
FORMAL DEFINITIONS FOR SPATIAL PREDICATES ON MERS

SERs. In addition, assume that r (or R) denotes the left-hand side (LHS) operand of the operation represented in infix notation (e.g., r `vertex_inside` r') and that r' denotes the right-hand side (RHS) one. Each element r of R partitions the space into the set of points belonging to the interior of r, denoted by r_{in}, the set of points belonging to the boundary of r, denoted by r_{on}, and the set of points not belonging to either r_{in} or r_{on}, denoted by r_{out}. Thus, $r = r_{in} \cup r_{on}$. A communication link between two closest boundary nodes of a region denotes a boundary segment if it does not intersect the interior of that region.

Two regions r and r' are `area_disjoint` if points lying in the interior of either region do not intersect any point of the other region, and neither region shares a minimal-unit region area with the other. For two regions that have an interior, the definition of `area_disjoint` given in [20] (viz., that no **point** lying in the interior of one of the regions intersects the interior or boundary of the other region) suffices. However, if either region is a unit region, the definition in [20] is not sufficient. Some example scenarios that show this are presented in Fig. 2. Thus, for `area_disjoint`, the definition has been extended to consider these cases. Two regions r and r' are `edge_disjoint` if they are `area_disjoint` and do not have any *common boundary segment* (CBS) although they may have one or more *common boundary node* (CBN). The `vertex_disjoint` relationship is simply **disjoint** if r and r' do not have any points in common. Two regions r and r' are `adjacent` if they are `area_disjoint` and have at least one CBS. r `meets` r' if r and r' are `area_disjoint` and having at least one CBN and no CBS. The `border_in_common` relationship is **true** of r and r' if they have at least one CBS. Region r is `area_inside` r' if r `equals` r' or r is a subset of r'. They may or may not have one or more CBN or CBS. Region r is `edge_inside` r' if r is `area_inside` r' and r and r' do not have a CBS. Region r is `vertex_inside` r' if r is `edge_inside` r' and r and r' do not have a CBN. r `intersects` r' if they share a common area. Formal definitions for spatial predicates over MERs is given in Tab. I.

V. TASK SPECIFICATION

A task is simple if it involves one spatial operation only, and complex otherwise. Boolean-valued tasks may also be composed using the **Boolean** connectives **and**, **or**, and **not**. We assume that a WSN is associated with a gateway that acts as the source for disseminating that tasks that express the desired analyses and as the sink for receiving the corresponding outcomes. The gateway is assumed to store information in its binding table about the regions that can be referred to in spatio-analytical tasks. The derivation of an interpretable structure in postfix notation from the task specification is performed at the gateway. This structure is then disseminated to the relevant nodes where it is evaluated by the task processing system with which each node is equipped. The gateway computes the task *minimum bounding rectangle* (MBR) and includes its coordinates as parameters in the task message. The task MBR denotes the rectangular region of the WSN in which the task needs to disseminated and evaluated. The MBR may comprise the whole of the WSN or, more likely, just a part of it. Tasks involving induced regions may need to be evaluated in every node as one cannot predict in advance how the corresponding phenomenon is to evolve. Consider the example in Sec.I. If the task is ($f5$ `area_disjoint` M) then the gateway sets the task MBR so that it encloses $f5$.

VI. TASK PROCESSING

The task processing can be broken down into three phases, viz., *task dissemination* (TD), *distributed task evaluation* (DTE) and *result processing* (RP).

A. Task Dissemination

In this phase, the task message is disseminated to the relevant nodes in the WSN. The task message conveys the following information: the interpretable form of the task, a pair of points that define the task MBR, the source node ID, the destination node ID, the evaluation period for the task, the length of time in which to (re)evaluate the task, and, finally, the time at the sink when it sent the task message. The task dissemination phase consists of two steps. In the first step, the task message is routed towards the MBR node that is closest to the gateway using greedy geographic routing [12]. In the second step, breadth-first broadcasting is used to disseminate the task message within the task MBR. The first task MBR node to receive the task message behaves in a special way: it takes on the role of *first-level leader*, records the source node ID and the destination node ID, updates the information in the packet related to the cost of reaching the source (i.e., the number of hops to the source) to zero, and sets its own ID as the source and destination node ID in the task message before broadcasting it. Upon receiving the task message, a task MBR node (other than the first) checks whether the cost of reaching the source is less than the one it has recorded earlier (this may be so because it may have already received the same task message from more than one neighbour). If it is, it records the source node ID as parent node ID, updates the cost of reaching source, and broadcasts the task message. If it is not, it ignores the message. Completing the two steps in the task dissemination phase accomplishes the additional purpose of electing the first-level leader and constructing the routing tree for result processing rooted on that node, to whom the results from leader nodes at the SER level are sent for aggregation, as described later.

B. Distributed Task Evaluation

Every task MBR node evaluates the interpretable structure conveyed by the task message, i.e., every node runs an interpreter for the geometry induction and algebraic evaluation steps required in our approach to spatial analysis. After receiving the task, a node may need to wait until the estimated time needed for the task message to be received by the furthest node in the MBR. In our approach, information about regions is distributed, across the nodes, a node belonging to one region does not have any information about other neighbouring nodes. Note that MERs may consist of any combination of SER with or without holes. The distributed task evaluation phase, in the case of **Boolean**-valued tasks, consists of two subphases, viz., *distributed membership evaluation* (DME) and *aggregation* (AGG).

1) Distributed Membership Evaluation.: The membership evaluation step is a process by which a node produces (on the basis of localized message-exchange only, and only in the worst case) a local outcome for each operation in the task. Tab. II codes the possible operation states into an octal digit (i.e., a 3-bit representation). An MBR node will declare its operation state to be **opNotApplicable** if the node does not belong to either of the operands of any of the **Boolean**-valued operators in the task. Otherwise, the node computes its membership state. For many operators (e.g., `vertex_disjoint`, `area_inside`), this only requires a node to do a look-up in its GIT. Some examples, now follow.

`vertex_disjoint` If a node belongs to the task MBR and to exactly one of the operands, it declares its operation state to be **true**, otherwise **false**.

`area_inside` A node declares its state to be **true** if it belongs to the interior or the boundary of r and to the interior of r', or to the boundary of both. The state is declared to be **unknown** if the node is a member of r' only. In any other case, it declares the state to be **false**.

`vertex_inside` A node declares its state to be **true** if it belongs to the interior or the boundary of r and to the interior of r'. The state is declared to be **unknown** if the node is a member of r' only. In any other case, it declares the state to be **false**.

In other cases (e.g., `adjacent`, `area_disjoint`, `edge_disjoint`), beside the local GIT look-up, CBNs require GIT information from their one-hop neighbours to compute their operation state. All non-CBNs can compute their operation state using the information available in their GIT. Some examples now follow.

`area_disjoint` All non-CBNs compute their operation state using the information available in their own GIT. If the node belongs to exactly one of the operands, then its operation state is **true**, otherwise it is **false**. Each CBN transmits an information request to its one-hop neighbours. Let the space around each CBN be divided into 12 sectors of 30° each. All one-hop neighbours that belong to one or both operands respond with their location and an indication as to whether they belong to the interior or the boundary of either operand. The CBN assigns the node, according to the responses, to sectors and finds out the closest neighbour in each sector.

OP state	Encoding	OP state	Encoding
true	1	**false**	0
disjoint	3	**unknown**	3
segment_node	5	**opNotApplicable**	7
CommonBoundaryNode	5		

TABLE II
REPRESENTING OPERATION STATES

Then, a CBN declares its state to be **false** if one or more of its neighbours belong either to the interior of both operands or to the boundary of one operand and the interior of the other. If a CBN finds no neighbouring non-CBN belonging to both regions, then it declares its state to be **true** if it has less than two neighbouring CBNs, otherwise it needs to find whether it is part of a common minimal-unit region. In this case, a CBN needs to consider the existence of a shared unit region. For this purpose, a CBN computes whether it forms a *common localized unit triangle* (CLUT) with its neighbouring CBNs. A *localized unit triangle* (LUT) $\langle\ s_1,\ s_2,\ s_3\ \rangle$ satisfies the properties that the interior and the edges of the triangle $\langle\ s_1,\ s_2,\ s_3\ \rangle$ do not contain any node that is a neighbour of s_1, s_2 or s_3, and all the edges have unit length (i.e., the prevailing radio range). If s_1, s_2, s_3 denote CBNs that form a LUT, then they form a CLUT. A CBN can compute this using the barycentric technique described in [3] for testing whether a point lies inside or on the boundary of a triangle. The order of the three CBNs in a CLUT is important. For each CLUT (in case a node participates in more than one), each CBN places itself and its neighbouring CBNs in an order based on their node ID. If a CBN finds that it belongs to one or more CLUTs, then it declares its operation state to be **false**, otherwise, **true**.

`adjacent` Nodes compute their membership in an `area_disjoint` relationship declare their state as **disjoint** if they are. For computing whether it forms a CBS, a CBN requires information from its one-hop neighbours as explained for the `area_disjoint` operation. In the case of `adjacent`, a CBN in a **disjoint** state makes use of the information it has received from its one-hop neighbours when computing `area_disjoint`. This information allows the CBN to construct the LUT graph of the subregion around it by taking itself as its origin (for each operand) [29]. Firstly, for the r operand (i.e., considering the LUT graph of the subregion formed by nodes belonging to r), the CBN computes whether the boundary segment formed with one of its CBNs is free, i.e., it is part of a LUT but no other LUT is adjacent to it. If it is free for r, it checks the same condition for r'. If the CBS is free for both r and r', it declares its operation state to be **segment_node**. If it finds that the boundary segment is shared by two adjacent LUTs, for either of the operands, the CBN repeats the process for the other CBNs. After repeating this process for its CBNs, if the node does not form a valid CBS, it keeps its initial **disjoint** state.

`meets` Nodes that are non-CBNs and are `area_disjoint` set their state to **disjoint**. A CBN with **disjoint** state makes use of the information it will have received from its neighbours as a result of verifying whether it is `area_disjoint` in order to compute whether it is part of a CBS. The CBN sets its state to **CommonBoundaryNode** if it is not part of a CBS, otherwise, it set its state to **false**.

$OP1_{Res}$	$OP2_{Res}$	OP for **and**	OP for **or**
false	**false**	**BitwiseAND**	**BitwiseOR**
!(**false**)	**false**	**BitwiseAND**	**BitwiseOR**
!(**false**)	!(**false**)	**BitwiseOR**	**BitwiseAND**
(**true**)	!(**true**)	–	**true**

TABLE III
BITWISE OPS FOR **AND** AND **OR** CONNECTIVES

`edge_inside` A non-CBN node sets its state to **true** if it is `area_inside`, and **unknown** if it is part of r' only. Otherwise, it set its state to **false**. A CBN needs to collect information from its neighbours in order to compute whether it forms one or more CBSs, as explained in `adjacent`. If it forms a CBS, it sets its state to **false**, otherwise to **true**.

`intersects` All non-CBNs compute their operation state using the information available in their own GIT: if the node belongs to exactly one of the operands, then its operation state is **unknown**, otherwise **true**. As is the case for `area_disjoint`, each CBN transmits an information request. All one-hop neighbours that belong to one or both operands respond with their location and an indication as to whether they belong to the interior or the boundary of either operand. Upon receiving the responses, a CBN assigns its neighbours to sectors and identifies the closest neighbour in each sector. If a CBN has neighbours in both regions that are not CBNs (e.g., on the boundary of one operand and in the interior of the other, or in the interior of both), then it declares its state to be **true**, otherwise it proceeds as follows. If it has less than two neighbouring CBNs, it sets its state to **unknown**. If it has two or more neighbouring CBNs it tests whether it is part of a CLUT. If it is, it sets its state to **true**, otherwise **unknown**.

`border_in_common` A node sets its state to **unknown** if it is not part of a CBS, otherwise to **true**. For this purpose, a CBN needs to request information from its neighbours as explained for `adjacent`.

Complex Task Evaluation.

This papers shows that the algorithms for topological operations maps the problem of distributed computation of complex algebraic expressions that involve multi-element geometries to the problem of first computing a node-level task state and then aggregating that at two levels using a new bit-string-based approach.

For the evaluation of complex tasks, a compression scheme is to merge the states for each operator into a node-level task state. Let a task have n operations, and let $[S_{OP_1}, S_{OP_2}, \ldots, S_{OP_n}]$, where each S_{OP_i} denotes an operation state in Table II, be a sequence of computed operation states. A node-level task state is generated by initializing it to 0 and then iteratively left-shifting its current value by three bits and **BitwiseOR**-ing S_{OP_i}.

Consider the example scenario in Fig. 3. Let the following task require evaluation:

```
(NOT(G VertexDisjoint h) AND (D Adjacent e)
   AND (G AreaInside D))
```

In Fig. 3, G and D are MERs, and h and e are SERs. The node labels indicate the task state, and arrow labels indicate the region-element task state resulting from aggregation.

Upon receiving the task message, all MBR nodes first evaluate `vertex_disjoint` operation, then the `adjacent` operation, and, finally, the `area_inside` operation.

The operation state and the task state (i.e., 120) is computed by node that lies in G only as follows. Since the node belongs to G only, it computes its operation state for (G `vertex_disjoint` h) as **true**. For (D `adjacent` e), it computes its operation state as **opNotApplicable**, as it belongs neither to D nor to e. For (G `area_inside` D), it computes its operation state as **false** since the node is a part of G but not of D. At the start, the node has task state **null**. The node acquires the task state 1 after assigning the first operation state (i.e., **true**) as the task state; for the second operation the task state (i.e., 1) is shifted three bits to the left and **BitwiseOR**-ed with 7 (i.e., **opNotApplicable**) to yield 15; and, finally, left-shifting 15 by three bits and **BitwiseOR**-ing the result with 0 (i.e., **false**) yields the task state 120.

2) Aggregation: The aggregation step processes node-level task states to yield a final outcome. This process of aggregation of partial results takes place at various levels. Firstly, at the level of individual region elements, for each element, node-level outcomes are partially aggregated into an SER-level outcome. Secondly, at the level of MER, the SER-level outcomes are aggregated into a first-level leader node (which has been determined during task dissemination as explained above) into the final aggregated result. After the computation of the node-level task state, the **BitwiseAND** operator is used to aggregate those task states.

The MBR nodes that participate in the aggregation are those with at least one operation state other than **opNotApplicable**. A node only needs to participate at most once in an aggregation process at the region-element level based on the following conditions. If the node has computed no operation state other than **opNotApplicable**, then it does not need to participate in the aggregation. If the node has computed exactly one operation state other than **opNotApplicable**, it participates in the aggregation on the basis of one of the operand (i.e., region) of the corresponding operation. The selection of operand is explained later in this section. If the node has computed more than one operation state other than **opNotApplicable** and belongs to more than one operand (i.e. region) involved, it participates in the aggregation but still on the basis of only one of those operand, as now explained. For participating in the aggregation, a node scans the operations in a complex task from left to right and selects the first operation for which it has computed a state other than **opNotApplicable**.

Once the operation is selected, the next step is to select the operand. It can be seen in Tab. I, that predicates `area_inside`, `edge_inside`, or `vertex_inside`, evaluate to true even if some elements in the right-hand side (i.e., R') operand contain more than one element from the left-hand side operand (i.e., R) or if some element of the R' operand does not contain any element of the R operand. For these operations, therefore, a node selects the R element. For the other operators, i.e., `vertex_disjoint`, `edge_disjoint`, `area_disjoint`, `meets`, `adjacent`, and `border_in_common`, any of the operands can be selected, so we select the left-hand side operand.

Aggregation is a well studied problem in WSNs [16]. A tree-based aggregation approach works by creating a tree rooted at a root node. The SER leader (i.e., the root) node can be

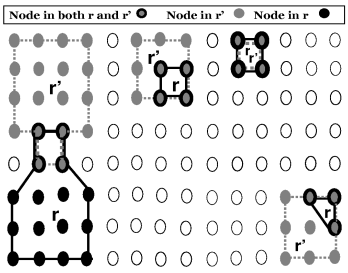

Fig. 2. Regions not `area_disjoint`

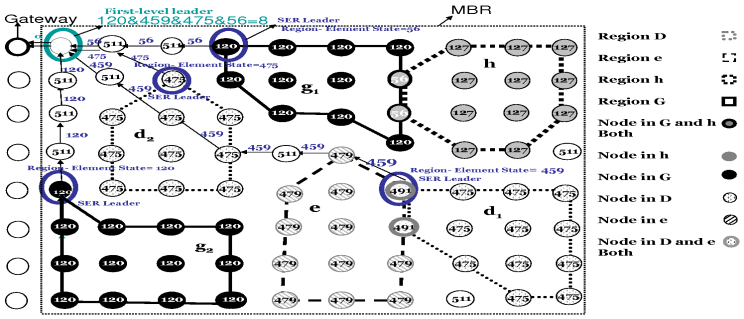

Fig. 3. Example scenario for evaluating complex task

selected randomly. We select as root node (i.e., SER leader) the node that is closest to the gateway. The tree construction phase is subdivided into two phases: *leader election phase* and *tree construction phase*. In the *leader election phase*, all nodes that are part of the relevant region broadcast their ID and operand ID (i.e., on the basis of which it participates in the aggregation process) to their neighbours. Upon receiving that information, each node computes whether there is a neighbour with a smaller ID. If there is, it sets its status as non-leader and sets a timer to wait for information from the leader node. When the waiting time expires, if has not received information about a leader node, it sends the *tree construction message (TCM)*. Otherwise, if it finds that it has the smallest ID, it starts the tree creation process by broadcasting a TCM. In which case, it sets the value of $leaderID$ and $sourceID$ attributes to its own ID, it sets the $level$ attribute to zero, the $distance$ attribute to its distance to gateway, and the $regionID$ to the selected operand ID.

When a node receives the TCM for the first time, it records the information obtained. It keeps the ID of the sender node as its parent node. After incrementing by one the $level$ attribute in the message and setting the $sourceID$ attribute to its own ID, the node broadcasts the message. If it receives a TCM from some other node, with a different $leaderID$ and a distance between this leader and gateway that is less or the same but with a smaller $leaderID$, than it updates its information, and broadcasts the message after updating attributes (as it is possible that more than one node declares itself a leader node). Otherwise, it ignores the message. If a leader node receives the TCM with better leader information, it records the information and changes its state to non-leader node. This process continues until all of the nodes in the region-element have been assigned a parent node, leader information and level.

Upon receiving the TCM, each node waits for a predetermined time which is reset when the TCM arrives from a neighbour. When the waiting period expires, each node registers with its direct parent node. This allows each node to learn how many child nodes it has. Nodes with zero children are leaf nodes. Once the tree has been set up, the process of aggregation of membership states works as follows. The process starts at the leaf nodes. Depending on its level and how many children it has, each parent waits for a period to receive the packets from its children. It then computes the partial outcome by applying the **BitwiseAND** operation to its own and the task states it received. It then forwards this partial outcome to its parent node. This continues at each level of the tree and eventually the task state at region-element level is computed at the SER leader node. Fig. 3, shows the region-element-level task state computed by each SER leader. There are four SER leaders transmitting the partial aggregated task states towards the *first-level* leader.

C. Result Collection

Once the distributed membership evaluation sub-task is concluded, results are routed towards *first-level leader* by the SER leaders along the tree that was established during task dissemination. The first-level leader performs the final aggregation (**BitwiseAND**) upon receiving the results from the SER leaders and computes the final aggregated task state. The first-level leader waits for certain period of time for the task states from SER leaders (there is more than one in the case of MERs). The first-level leader then decompresses the operation states in the final aggregated task state. These operation states denote the final outcome for the evaluation of each spatial predicate in the task. The next step is to apply the **Boolean** connectives (i.e., **not**, **and** and **or**) that occur in the task. For this purpose, each spatial predicate in the task expression is replaced by the final outcome computed for it and the **Boolean** connectives are applied (using the classical Boolean algebraic rules) to compute the final outcome as either **true** or **false**. If application of connectives results in a **true**, then final outcome is declared as **true**, otherwise **false**. This is then transmitted by the first-level leader towards the gateway. Tab. III defines how **and** and **or** correspond to bitwise operations depending on the operations states they apply to. In Tab. III, !(**false**) denotes any operation state in Tab. II other than **false**, and !(**true**) denotes any operations state other than **true**. **not** yields **true** if the operation state is any other than **opNotApplicable** and **true**. In Fig. 3, the *first-level* leader, after applying the **BitwiseAND** aggregation operator to the partial outcomes received from the SER leaders, computes the MER task state as 8. It then decompresses and retrieves the operation states (e.g., $8 = 000001000_2$) yields **false**:**true**:**false**. It then interleaves the connectives as stipulated in the linearized task expression to yield **not** (**false**) **and** (**true**) **and** (**false**)). When this Boolean expression is evaluated it yields the final task value **false**.

Recall, from the formal definitions in Table I, that a `vertex_disjoint` relationship between MERs holds only if it holds for all the elements in both regions. In the example scenario, region G has two elements $g1$ and $g2$

but `vertex_disjoint` only holds for $g2$. Therefore, final aggregated state for `vertex_disjoint` operation is **false**. $d1$ is `adjacent` to e, but $d2$ is not. $d2$ is only `area_disjoint` with e. Therefore, the final aggregated state for `adjacent` is **true**. In the case of `area_inside`, no element of G lies within one or more region elements of D. Therefore, the final aggregated result for `area_inside` is **false**.

VII. Experimental Evaluation

This section presents experimental evidence as to the efficiency of our framework in simulated, but realistic, deployment scenarios (explained in Sec. I). The experiments provide evidence that the algorithms scale well in terms of message complexity, energy consumption and response time. At the time of writing, as Sec. II describes, there is no comparably expressive, implemented platform for in-network spatial analysis that we could compare our implementation with. We therefore compared our approach with an out-of-network approach in which boundary information of induced regions are sent back to the gateway. In all the experiments presented in this section, it is assumed that the tasks related to detection of induced region is run over the whole network. Because of the space constraint, we have not included the cost of boundary detection in the experimental costs [9]. Also, we note that the comparison is reproduced here as it was first published in [10] in order to make this paper more self-contained.

Experimental Set-Up:

We have implemented all the operations in the algebra in nesC/TinyOS. For performance evaluation, we use PowerTOSSIM [21], a power modelling extension to TOSSIM that provides a per-node estimation of power and scales to larger networks. The specification of the nodes we have simulated is [Type = Mica2, Radio = CC1000, Energy Stock = 31,320,000 mJ (2 Lithium AA batteries)]. We used TinyViz for modelling the behaviour of our network. We have written several pluggins to make each node aware of specific information (e.g., location, information about asserted regions etc.). We have set the radio connectivity based on distances between nodes and therefore, the maximum cardinality of the one-hop neighbourhood of a node is set to eight in all our experiments. For energy efficiency, all MBR nodes enter a low-power listening state [18], [14] after the completion of distributed membership evaluation. This allows them to listen to requests to route packets if required. The motivation is the fact that only SER leaders (and relay nodes) are needed to route packets and it is not known in advance which nodes will become SER leaders. In routing the results, we have implemented a simple protocol that makes a fixed number of attempts to transmit the message to the parent about which it recorded information in task dissemination phase.

Experiment 1: Behaviour w.r.t. Network Growth This experiment explores the behaviour of the implemented algorithms as, given a task, the number of nodes in the network grows. The task we use is one that, in terms of our motivating example, identifies whether spraying is required in a given field. Fig. 4 shows field `f5` and two induced regions. The task used in experiment is:

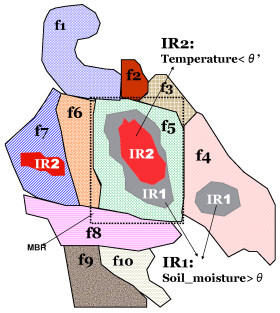

Fig. 4. Distinct Intersection Induced Regions

```
(NOT ((IR1 AreaDisjoint f5) AND (IR2 AreaDisjoint f5)
                            AND (IR2 VertexDisjoint IR1)))
```

Our aim is also to convey an impression of the richness of the proposed framework and algebra and, therefore, we have expressed the task in such a manner as to cover more of the available operators. The same task can be expressed as `((IR1 intersects f5)` **`and`** `(IR2 intersects f5)` **`and`** `(IR2 intersects IR1))`. The results obtained are depicted in Fig. 5. Note that in this figure, and similar ones, the results are broken down into four components corresponding to the evaluation phases, viz., TD for task dissemination, DME for distributed membership evaluation, TC+AGG for tree construction and aggregation, RP for result processing. The time required for aggregation and routing at two-levels is split into two parts, viz., AGGSER, i.e., the amount of time taken to construct tree and aggregation at level of SERs, and AGGMER, i.e., the time taken to compute the final result at the first-level leader. Because the first-level leader waits for a response from all the SERs before computing the final result, to avoid overlap, in Fig. 5**c**, the sum (AGGMER + RP) denotes the time taken from the aggregation being completed at the last SER until the result being received at the sink. The size of the regions is given in Tab. IV.

Our implementation exhibits slow rates of growth in terms of messages transmitted, energy consumed and response time as the network size grows. As expected, the most onerous phases are TD and AGG, as they are responsible for the majority of communication events. The energy consumption seems to exhibit particularly good behaviour. For the largest MBR (containing 120 nodes), the amount of energy consumed by CPU and radio together is around 138,481 mJ. If each node is powered by two AA batteries, the initial energy stock per node is 31,320,000 mJ. The total energy stock inside the MBR is then 120*31,320,000. This implies that each evaluation episode consumes less than 0.004% of the total energy stock. Even if the depletion is not uniform across different nodes, the overall amount depleted is very low. Low enough that adding the energy required to induce the event regions (not counted in Fig. 5(**b**)) is unlikely to significantly detract from the force of these conclusions.

Experiment 2: Behaviour w.r.t. Dynamic Evolution: This experiment explores the behaviour of the implemented algorithms as, given a network with 223 nodes, a transient physical phenomenon evolves through six stages. In terms of our motivating example, we note that water infiltration varies over

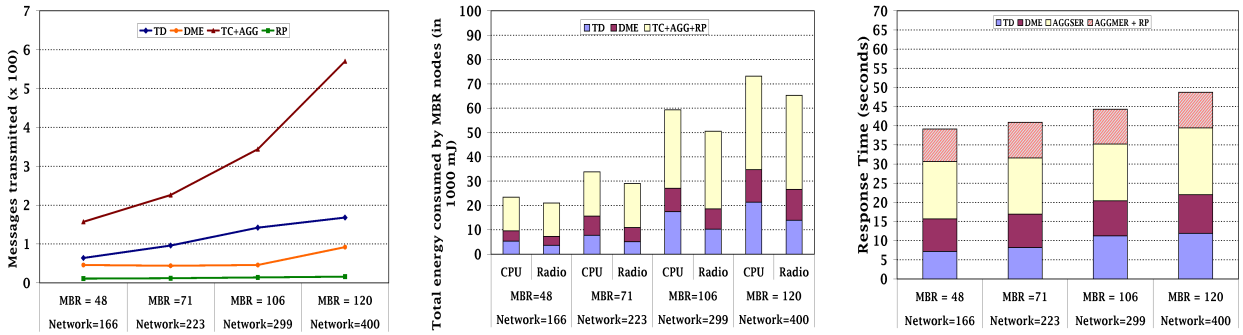

Fig. 5. Exp. 1: (a) Messages Transmitted, (b) Energy Consumed And (c) Response Time, As The Network Grows

space and time and, hence, influences soil moisture levels differently across monitoring episodes.

In Fig. 6 an induced region `IR1` undergoes six episodes of change: at Eval1, it is a SER comprising 15 nodes; at Eval2, it grows to 24 nodes and has moved; at Eval3, it grows to 37 nodes and changes shape; at Eval4, it splits and becomes a MER, with one region comprising 14 nodes and another comprising 31 nodes; at Eval5, one of the component elements has developed a hole, with one of the elements comprising 14 nodes, and the other comprising 27 nodes and containing a hole consisting of 05 nodes; at Eval6, one of the component elements has grown to 35 nodes, as has the hole inside it, from 05 to 10 nodes. In these experiments `f5` comprises 63 nodes and `f6` 35 nodes. The task used in the experiment is:

```
((IR1 Intersects f5) AND
   ((IR1 BorderInCommon f6) OR (IR1 AreaInside f6)))
```

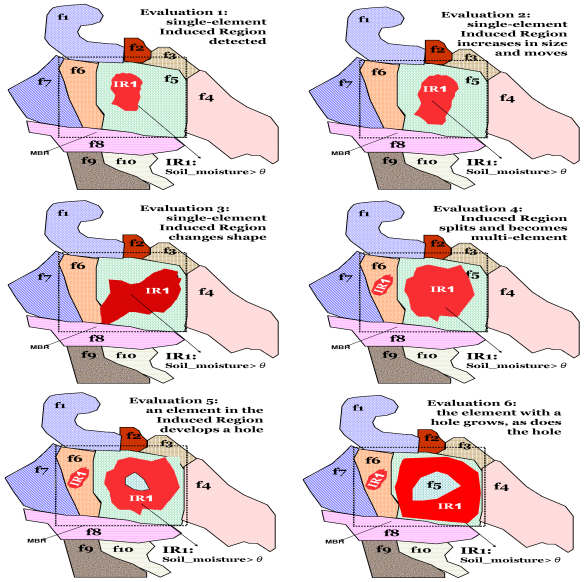

Fig. 6. Six Stages of an Evolving Phenomenon

The results obtained are depicted in Fig. 7. Again, our implementation exhibits slow rates of growth in terms of messages transmitted, energy consumed and response time as the regions that characterize an evolving phenomena change over time. The increase in complexity, and the sparser, more spread-out nature of the region element that contains a hole, as expected, implies a more onerous AGG phase in terms of messages transmitted. However, as Fig. 7(**b**)-(**c**) show, there is very little variation in energy consumption and response time across the evaluation episodes. This is satisfying, as it suggests that our approach is stable in terms of energy consumed even as a transient phenomenon undergoes significant topological change.

Experiment 3: In-Network v. Out-of-Network Processing: This results from this experiment first appeared in [10]. We reproduce them here to make the paper more self-contained. The purpose in this case was to measure the cost in terms of messages transmitted, energy consumed and response time of sending all the event information from boundary nodes back to the gateway for processing outside the network. It explores the extent to which in-network processing offers savings w.r.t. out-of-network processing. The experiment is run over the scenario as shown in Fig. 4. In this experiment, it is assumed that each node knows the parent to which it has to transmit its message and the messages received from its children. A node tries up to four times to send a packet, otherwise, it broadcasts, and also checks not to send duplicate packet.

Tab. VII shows how, in the case of both in- and out-of-network processing, the messages transmitted, the energy consumed and the response time grow as the network size grows. So, to allow for comparison, in Tab. VII we have used the data (in Fig. 5) from Experiment 1 above. In case of out-of-network processing, for the network containing 166 nodes the amount of energy consumed by CPU and radio together is close to 422 kmJ. If each node is powered by two AA batteries. This implies that each evaluation episode consumes between 0.013% and 0.087% of the total energy stock for networks between 166 and 400 nodes, respectively. However, note that the distribution of energy consumption is not uniform: nodes that are closer to the sink will deplete their energy stock much earlier. It can be seen that all costs for out-of-network processing are very high compared to our in-network implementation. Our implementation transmits no more than half the messages, and often much less; it consumes less energy and requires less time to respond.

VIII. Conclusion

This paper has described a framework for representing asserted, induced, and derived regions in WSNs as well as

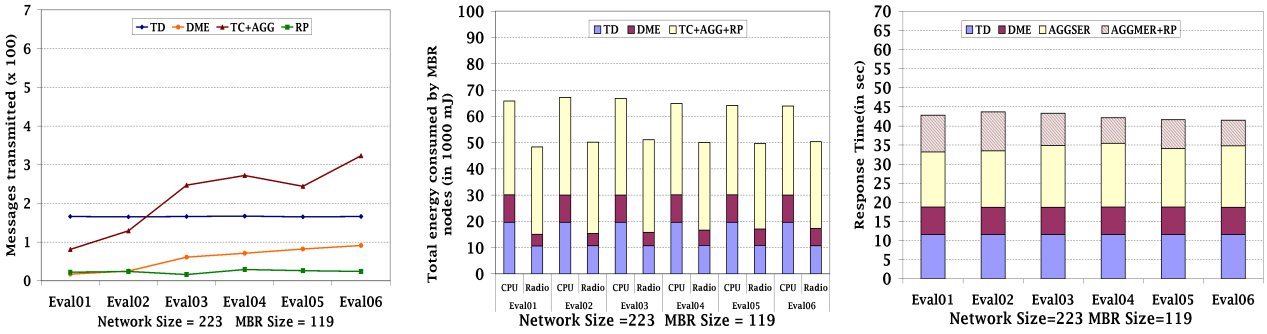

Fig. 7. Exp. 2: (a) Messages Transmitted (b) Energy Consumed And (c) Response Time, As The region Evolves

Network Size	f5	IR1 in f5	IR2 in f5	IR1 out f5	IR2 out f5
166	48	28	11	12	6
223	63	37	15	18	10
299	86	51	20	22	14
400	106	68	28	27	18

TABLE IV
SIZE OF REGIONS IN EXP. 1

Nodes	Bytes (*100)		Messages (#)		Energy (kmJ)				Resp. (sec)	
	In	Out	In	Out	In		Out		In	Out
					CPU	Radio	CPU	Radio		
166	30	73	278	1175	23	21	245	177	39	140
223	41	121	378	1902	34	29	472	357	41	197
299	60	157	546	2515	59	51	1032	646	44	249
400	81	237	846	3758	73	65	1496	835	49	306

TABLE V
EXP. 3: IN- VS. OUT-OF-NETWORK PROCESSING

topological operations on the latter inspired by the Schneider-Güting ROSE algebra [20] over the regions representable by the framework. We have described an implementation of the algebra for in-network processing in WSNs and demonstrated that it is efficient. The algorithms are, for the most part, localized, and were shown by our experiments to have very desirable behaviour in terms of message complexity and energy consumption. Our results show that in-network evaluation of spatial analytic tasks results in substantial savings in terms of energy consumption and response time (as well as, in message and bit complexity) compared to sending all boundary information of transient phenomena back to the sink. Future work would benefit upon these contributions to enable high-level specifications of expressive spatial analysis over WSNs.

IX. ACKNOWLEDGEMENT

Farhana Jabeen, thanks the Schlumberger, for their support under their Faculty for the Future program.

REFERENCES

[1] Xianjin Zhu and Rik Sarkar and Jie Gao and Joseph S. B. Mitchell. Light-weight Contour Tracking in Wireless Sensor Networks. In *INFOCOM*, 2008.
[2] P. Bonnet, J. E. Gehrke, and P. Seshadri. Querying the physical world. *IEEE Journal of Selected Areas in Communications*, 7(5):10–15, Oct. 2000.
[3] H. Coxeter. *Introduction to Geometry*, pp. 216–221. Wiley, 1969.
[4] M. J. Egenhofer, E. Clementini, and P. D. Felice. Topological relations between regions with holes. *IJGIS*, 8(2):129–142, 1994.
[5] C. Farah, C. Zhong, M. F. Worboys, and S. Nittel. Detecting topological change using a wireless sensor network. In *GIScience*, pp. 55–69, 2008.
[6] I. Galpin, C. Y. A. Brenninkmeijer, F. Jabeen *et al.* Comprehensive optimization of declarative sensor network queries. In *SSDBM*, pp. 339–360, 2009.
[7] T. Griffiths, A. A. A. Fernandes, N. W. Paton *et al.* Tripod: A comprehensive system for the management of spatial and aspatial historical objects. In *ACM-GIS*, pp. 118–123, 2001.
[8] J. K. Hart and K. Martinez. Environmental Sensor Networks: A revolution in the earth system science? *Earth-Science Reviews*, 78:177–191, 2006.
[9] F. Jabeen and A. A. A. Fernandes. Impact on accuracy of deployment trade-offs in localized sensor network event detection. In *LOCALGOS, in conjuction with IEEE DCOSS*, 2008.
[10] F. Jabeen and A. A. A. Fernandes. Monitoring spatially-referenced entities in wireless sensor networs. In *MENS, in conjunction with UIC-ATC'10, IEEE*, 2010. To appear.
[11] J. Jiang and M. F. Worboys. Detecting basic topological changes in sensor networks by local aggregation. In *GIS*, page 4, 2008.
[12] B. Karp and H. T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *MOBICOM*, pp. 243–254, 2000.
[13] B. Koch and R. Khosla. The Role of Precision Agriculture in Cropping Systems. *Journal of Crop Production*, 9(1/2 (17/18)):361–381, 2003.
[14] O. Landsiedel, K. Wehrle, and S. Gotz. Accurate prediction of power consumption in sensor networks. In *EmNets*, pp. 37–44, 2005.
[15] C.-H. Lin, C.-T. King, and H.-C. Hsiao. Region abstraction for event tracking in wireless sensor networks. In *ISPAN*, pp. 274–281, 2005.
[16] S. Madden, M. J. Franklin, J. M. Hellerstein *et al.* TAG: A Tiny AGgregation Service for Ad-Hoc Sensor Networks. In *OSDI*, 2002.
[17] S. Madden, M. J. Franklin, J. M. Hellerstein *et al.* TinyDB: an acquisitional query processing system for sensor networks. *ACM Trans. Database Syst.*, 30(1):122–173, 2005.
[18] U. Malesci and S. Madden. A measurement-based analysis of the interaction between network layers in TinyOS. In *EWSN*, pp. 292–309, 2006.
[19] L. Mottola and G. P. Picco. Logical neighborhoods: A programming abstraction for wireless sensor networks. In *DCOSS*, pp. 150–168, 2006.
[20] M. Schneider. *Spatial Data Types for Database Systems*, *LNCS* 1288, 1997.
[21] V. Shnayder, M. Hempstead, B. Chen *et al.* Simulating the power consumption of large-scale sensor network applications. In *SenSys*, pp. 188–200, 2004.
[22] T. Ulrich. Wireless network monitors h2o: System saves resources, increases yield in cabernet vineyard. *Wines and Vines Magazine*, July 2008.
[23] M. Welsh. Exposing resource tradeoffs in region-based communication abstractions for sensor networks. *Computer Communication Review*, 34(1):119–124, 2004.
[24] M. F. Worboys and M. Duckham. Monitoring qualitative spatiotemporal change for geosensor networks. *IJGIS*, 20(10):1087–1108, 2006.
[25] Y. Yao and J. Gehrke. The cougar approach to in-network query processing in sensor networks. *SIGMOD Record*, 31(3):9–18, 2002.
[26] Budhaditya Deb and Sudeept Bhatnagar and Badri Nath. STREAM: Sensor Topology Retrieval at Multiple Resolutions. In *Kluwer Journal of Telecommunications Systems*, vol 26, pp. 285–320, 2003.
[27] Kun Bi and Naijie Gu and Kun Tu and Wanli Dong. Neighborhood-based distributed topological hole detection algorithm in sensor networks. In *IET Conference Publications*, vol 26, pp. 285–320, 2006.
[28] K. Chintalapudi and R. Govindan. Localized edge detection in sensor fields. *Ad Hoc Networks*, 1(2-3):273–291, 2003.
[29] Romulo G. Pizana, Roland E. Ramos. Triangle graphs with maximum degree at most 3. *Proc. of the Third Asian Mathematical Conference*, pp. 451–454, 2000.