

Monitoring Spatially-Referenced Entities in Wireless Sensor Networks

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

Abstract—Wireless sensor networks (WSNs) make it possible to gather information about the physical world in novel ways and in unprecedented levels of detail. In the environmental sciences, in particular, WSNs are on the way to becoming an important tool for monitoring spatially- and temporally-extended physical phenomena. However, support for high-level and expressive spatio-analytic tasks to explore relationships between spatially-referenced entities (e.g., whether mist is over a vineyard or has not yet touched it) and to derive representations grounded on such relationships (e.g., the geometrical extent of that part of a vineyard that is covered by mist) is still incipient, particularly in the case of in-network processing approaches that are imperative, due to energy scarcity, in mote-level WSNs. This paper describes distributed implementations of a comprehensive collection of spatio-algebraic operations that enable the expression and evaluation of tasks involving spatial predicates as well as the derivation of new geometries from existing geometries. This makes it possible to report back only fine-grained information.

I. INTRODUCTION

WSNs allow for interaction with the environment at very high spatial and temporal densities. Since each sensor node has a location in physical space, WSNs provide an interface to the physical world and enable us to associate spatial properties with sensed data. Since WSNs can evaluate tasks periodically, we can also associate temporal markers with sensed data. This can be crucial in many applications scenarios, e.g., in environmental monitoring [7], precision agriculture [12], etc.. WSNs may be viewed as a distributed computing platform [2], with each node being viewed as computational resource and not just a data collection and data transmission resource. This paper takes this view and contributes techniques for periodic, in-network evaluation of spatio-analytic tasks.

Precision agriculture is a developing discipline aiming to enhance farming efficiency [12]. One real-world example in which WSNs are being deployed is at Camalie Vineyards [19]. As an example of the usefulness of WSNs in this area, consider the following context. Imagine that, for efficient water management, a grower has deployed sensor nodes, and is interested in identifying whether, in some part of a field, soil moisture has dropped below a certain threshold, so that only those parts are irrigated, given the limited water supply. In addition, the grower needs to decide whether to apply pesticides (based on whether certain temperature and humidity conditions are met). It can be seen that much of the information required relates to location, as the factors affecting crop quality are inherently spatial in nature. As the basis for our examples and experiments, we have used the underlying geometries at the

Camalie Vineyards deployment¹ (see Fig. 3). Our motivation is to enable WSNs that allow the expression of information requirements as spatio-algebraic expressions whose evaluation returns fine-grained, high-content information. More formally, given thresholds θ and θ' , call IG1 the event geometry within which soil moisture is above θ and IG2 the event geometry within which temperature is below θ' . In this case, the task whose evaluation determines whether there is a need to spray a field, say $f5$, against fungal disease is:

$$\text{(NOT ((IG1 AreaDisjoint f5) AND ((IG2 AreaDisjoint f5) AND (IG1 VertexDisjoint IG2))))}$$

Assume that field $f5$ comprises two types of soils, ST_1 and ST_2 . Call SA the area where soil is of type ST_1 and SB the area where soil is of type ST_2 . One can then derive a new geometry that characterizes the area with soil type ST_1 in field $f5$ where neither soil moisture nor temperature are above the target thresholds by posing the task:

$$\text{((F5 Intersection SA) Minus (IG1 Plus IG2))}$$

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., *area_disjoint*, *vertex_disjoint*). In this paper, we focus on spatial-valued operations. Boolean-valued operations are the focus of [9].

The remainder of the paper is structured as follows. We discuss related work in Sec. II. Sec. III describes our framework for representing geometries in WSNs. Sec. IV describes our proposed algebra for geometry derivation. Sec. V describes task specification. Sec. VI describes the distributed algorithmic strategy used in the evaluation of spatial-valued tasks. An empirical evaluation of the algorithms in Sec. VI is presented in Sec. VII. We conclude in Sec. VIII.

II. RELATED WORK

The work described in [21] provides a computational model for WSNs to detect spatial change in dynamic regions based on local, low-level snapshots of spatio-temporal data. Other work on using WSNs to detect topological change (i.e., hole formation, hole loss, splitting and merging of event region) includes [4], [10]. Three boundary detection methods are

¹More information is available at <http://camalie.com/WirelessSensing/WirelessSensors.htm> [Accessed: 7 Aug 2010]. In particular, see <http://camalie.com/CamalieGIS/Naked/> [Accessed: 7 Aug 2010] for the geometries.

studied in [23]. Their evaluation approach involves sending boundary information to the gateway, so that it can generate the snapshot of the event region. In [19], the sensed measurements are transmitted towards the gateway to generate a snapshot of the event region. The framework proposed in [13] dynamically builds a structure over the nodes part of the event region for information collection and aggregation. The focus in [13] is on functions like average, min, max, etc.. The work reported in [20] focuses on providing a high-level programming interface that abstracts away the details of routing, data collection of routing, data dissemination, and state management. In [22], the authors propose an algorithm for the retrieval of topology at multiple resolutions.

None of the above proposals takes an algebraic approach like we do. This suggests that their expressiveness is likely to be narrower than ours, since they are not inherently compositional due to lack of closure. Our algebraic approach is also aimed at easing the process by which our operations could be embedded in a declarative query language.

Our algebra is closely inspired by the ROSE algebra [17]. Schneider and Güting developed the ROSE algebra for implementation in spatial database systems (one such implementation is described in [6]). The ROSE algebra supports points, lines, and regions on a plane. Over such data types, a comprehensive set of algebraic operations is defined. The ROSE algebra has several desirable characteristics for the purposes of the work described in this paper: (i) it rigorously defines a comprehensive set of Boolean-valued and spatial-valued operations; and (ii) it supports complex geometries such as regions containing holes, and multi-element ones. However, the centralized algorithms in [17], 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.

III. SPATIAL FRAMEWORK FOR WSNs

In this paper, to a WSN there corresponds a finite, discrete, two-dimensional space that models a spatial domain. Let $M \times N$ denote an Euclidean plane that discretizes a rectangular 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 determines its position in the WSN based on its location. We assume that the sensing and communication range of a node $s_i \in S$ is representable by a circle with radius $r_{s_i}^d$ and $r_{s_i}^c$ 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. This allows geometries to be defined as subgraphs of the WSN connectivity graph. Following the ROSE-algebraic approach [17], application-specific geometries can be defined based on abstract data types such as **points** (denoting, e.g., a well), **lines** (denoting, e.g., a river), **regions** (denoting, e.g., a cultivated field). Note also that there may or may not be a node in every point in the grid, but, in our framework, spatial values are always represented in terms of deployed nodes. A

node can be part of one or more geometries. Each sensor node stores and manages geometry-related information locally in a *Geometric Information Table (GIT)*. An entry in a *GIT* is a quadruple $\langle GID, DT, BN, TTL \rangle$, where *GID* (for *geometry ID*) is a unique identifier for the corresponding geometry, *DT* denotes the type of the geometry, *BN* is **true** iff s_i is in the boundary of the geometry, and *TTL* denotes the time-to-live after which the *GIT* entry becomes invalid.

Our framework supports three kinds of geometries, viz., asserted, induced and derived. The asserted geometries are representations of permanent physical features (e.g., a well, a river, or a cultivated field). These geometries are assumed to both pre-exist the WSN deployment and to remain unchanged during that deployment. Therefore, the *TTL* for asserted geometries is set to ∞ and is never updated. Induced geometries are representations of transient physical phenomena (e.g., an area of mist). They are assumed not to pre-exist the WSN deployment. These geometries are characterized by means of event detection and boundary computation. Briefly, by *event detection* we mean the outcome of a distributed process in which participating nodes evaluate an *event-defining predicate* (e.g., `temperature>10`) and, as a consequence, may (if the predicate is satisfied) declare themselves *event nodes*. The boundary of the event geometry can then be computed through the distributed boundary detection algorithms [8]. The *GID* is provided as part of task message. The entry in *GIT* for induced geometries is maintained dynamically, as follows. If the *GID* already exists, its *TTL* is reset (typically, in the case of periodic evaluation, until the next evaluation period). Otherwise, a new entry is added. When its *TTL* expires, the entry is removed. In this paper we describe how to compute derived geometries and how information about such geometries is maintained by the nodes. The *GIT* allows a node to keep record of which geometries it is part of. This is used, e.g., to decide whether the node should evaluate some task at a specific evaluation period.

The proposed framework not only allows for the periodic detection of induced geometries representing evolving phenomena, it also supports periodic derivation of geometries and evaluation of spatial predicates. As in [15], the evaluation period (e.g., every 30 minutes) and duration are parameters. Energy efficiency is made likely by the fact that information about geometries is kept inside network, thereby avoiding the need to ship all the information to the gateway: only fine-grained answers are provided to the user.

IV. SPATIAL ALGEBRA FOR THE DERIVATION OF GEOMETRIES OVER WSNs

There are three spatial types, viz., **points**, **lines**, and **regions**. A **points** value denotes a (possibly singleton) finite set of nodes with location (x, y) in the finite, discrete, two-dimensional sensor space defined by a WSN deployment as described in Sec. III. We denote the location (x, y) of a node by writing the node ID, e.g., s_1 . A *line segment* is a pair of distinct locations s_1-s_2 in sensor space, such that s_1 and s_2 are closest one-hop boundary neighbours. A **lines** value is a (possibly singleton) finite set of pairwise disjoint, *line segment* values forming a connected sequence with no interior.

A **regions** value is (possibly singleton) finite set of triples of the form $\langle b, i, H \rangle$ where b is a *cycle* value denoting the *boundary* of a region, i denotes the *interior* of the region and is possibly empty set that contains all the node IDs enclosed by b excepting those belonging to H , where H is a possibly empty set of disjoint *cycle* values lying in the interior of b , each element of which denotes a hole in the region. Thus, the boundary and the exterior of a **regions** value are allowed to be disconnected. Therefore, a **regions** value with a hole has one outer boundary and one or more interior boundaries depending on the number of disjoint holes inside it. Each inner boundary defines one hole in the **regions** values. Note that the nodes enclosed by any element of H do not belong to i , and hence not to the **regions** value either. As will be seen later, a node only keeps information about a geometry if it lies in the interior or in a boundary of that geometry. A **regions** value r partitions the space into the (possibly empty) 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} . In the case of **regions** value with holes $r_{on} = r_{oon} \cup r_{ion}$ i.e., the boundary of r is the union of outer and inner boundaries. Thus, $r = r_{in} \cup r_{on}$. A *unit regions* value is a (possibly singleton) finite set of pairwise disjoint, cycles having an empty interior. Nodes that form a *unit regions* value represent a boundary. A *minimal unit regions* value is a smallest *unit regions* value, i.e., a (possibly singleton) finite set of three-node cycles (i.e., triangles) having an empty interior. If a non-empty value of type **points**, **lines**, or **regions** is a singleton we refer to it as a *single-element geometry* (SEG), otherwise as a *multi-element geometry* (MEG).

The signatures for spatial-valued operations are given in Table I. Due to space constraints, in Table II we only provide formal definitions for these operations when the arguments are **regions** values (these adapt and extend those in [17]). In Table II, r and r' denote SEG **regions** values with or without holes; s , s' and s'' and s''' denote node IDs that are closest one-hop neighbours; $\overline{ss'}$ denotes a segment between s and s' ; $\overline{ss'}$ denotes that s and s' are boundary neighbours and form a boundary segment that does not intersect the interior of a **regions** value; r_{on} and r_{in} are as before. Let $s \in r_{on}r'_{on}$ denote that s belongs to the boundary of both r and r' , and let $s \in r_{in}r'_{on}$ denote that s belongs to interior of r and the boundary of r' . Let $MinUnit(r)$ denote the *localized unit triangle* (LUT) r . A LUT $\langle s_1, s_2, s_3 \rangle$ satisfies the properties that the interior and the edges of the LUT do not contain any node that is a neighbour of s_1, s_2 or s_3 and that all the edges of LUT have unit length (i.e., the prevailing radio range). Let $MinUnit(rr')$ denote a common localized unit triangle of r and r' and $MinUnit(r_{on}r'_{on})$ denote a common localized unit triangle of r and r' formed by boundary nodes.

V. TASK SPECIFICATION

In our design, we assume that a WSN is associated with a gateway that acts as the source for disseminating tasks expressing the desired analysis and as the sink for receiving the corresponding outcomes. The gateway is also assumed to

store information about the geometries 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 and gives rise to an interpretable structure that is evaluated by the task processing system with which each node part of the network is equipped with. 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.

VI. TASK EVALUATION

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

Task Dissemination Along with other required attributes, the task message conveys the interpretable form of the task, a pair of points that define the task MBR, the derived geometry ID, the source and destination node IDs and the number of hops taken. 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 [11]. 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*, it records the information, updates the number of hops to zero, sets its own ID as the source and destination node ID in the task message, and then broadcasts it. When a node in the MBR receives the task message, it records the information in it and broadcasts it. Task dissemination ultimately defines a routing tree for partial result aggregation and final result collection.

Distributed Task Evaluation Each node is equipped with the task processing system, which uses a stack-based approach to evaluate the task expression. Spatial-valued task evaluation is a two-step process comprising membership evaluation and geometry derivation. *Membership evaluation* is the process whereby a node computes whether it satisfies the conditions for membership in the derived geometry defined by the spatial-valued operator. *Geometry derivation* is the process whereby a node that satisfies the membership conditions computes it lies in the boundary or interior of the derived geometry. A node may also skip the above (declaring the operation not applicable to it) if it is in the task MBR but does not belong to the operand(s) involved or if the geometries it participates in are not of the type of the operand(s) involved. We now give details on how the operators involved in our examples are evaluated.

Plus In the case of the **plus** operator, if an MBR node finds itself part of one or both operands, it declares itself part of the derived geometry (i.e., an event node) by setting its operation state to **true**. A *common boundary node* (CBN) must perform localized decision-making to compute their edge state. A CBN requests information from its neighbours that have a **true** operation state. Upon reception of the answers, a CBN node makes itself the origin of a circle centered at itself and

Intersection	: Points × Points	→	Points
Intersection	: Lines × Lines	→	Points
Intersection	: Regions × Regions	→	Regions
Intersection	: Regions × Lines	→	Lines
Contour	: Regions	→	Lines
Plus	: Points × Points	→	Points
Plus	: Lines × Lines	→	Lines
Plus	: Regions × Regions	→	Regions
Minus	: Points × Points	→	Points
Minus	: Regions × Regions	→	Regions
Minus	: Lines × Lines	→	Lines
Vertices	: Lines	→	Points
Vertices	: Regions	→	Points
CommonBorder	: Lines × Lines	→	Lines
CommonBorder	: Regions × Regions	→	Lines

TABLE I
SPATIAL-VALUED OPERATIONS

partitions its neighbouring event nodes as lying on a quadrant or a axis based on their location. It then uses the modified T-Fit [8] boundary detection algorithm to compute its edge state. A boundary node with **true** operation state that belongs to only one operand sets its edge state **true**, otherwise **false**.

Intersection In the case of the `intersection` operator, non-CBNs compute their operation state using the information available in their own *GIT*: if the node belongs to the interior of both operands or to the interior of one and the boundary of the other, or to boundary of one and the interior of the other, then its operation state is **true**, otherwise **false**. Let the space around a CBN be divided into 12 sectors of 30° each. A CBN needs to transmit an information request to its one-hop neighbours. All one-hop neighbours that belong to one or both operands respond with their location and an indication as to whether they belong to interior or boundary of each of the operands. Upon receipt of this information, a CBN assigns them to the proper sector (based on their location) and computes the minimum distance neighbour in each sector. Its operation state is **true** if one or more of its closest one-hop neighbours nodes belong either to the interior of both operands or to the boundary of one operand and the interior of the other. If a CBN has no neighbouring non-CBN belonging to both geometries, then its operation state is **false** provided that it has less than two neighbouring CBNs, otherwise it must consider the existence of a shared minimum unit **regions** value. Let s_1, s_2, s_3 denote CBNs then a LUT between them may form a *common localized unit triangle* (CLUT). A CBN can compute whether this is the case using the barycentric technique in [3]. The order of 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 itself part of one or more CLUTs, its operation state is **true**, otherwise **false**. A node with **true** operation state that does not belong to the interior of both operands sets their local edge state attribute to **true**, otherwise to **false**.

Minus Let the operation be $r \text{ minus } r'$. A non-CBN node that belongs to r only or to the interior of r and the boundary of r' , sets its operation state to **true**, otherwise to **false**. As for intersection, in this case too a CBN must consider CLUTs. Let the space around a CBN be divided into 12 sectors of 30° each. A CBN needs to transmit an information request to its one-hop neighbours. Upon receipt of this information, the CBN assigns them to the proper sector (based on their

$r \text{ Plus } r'$	\equiv	$\{s \mid s \in r \vee s \in r'\}$
$r \text{ Intersection } r'$	\equiv	$\{s \mid s \in r_{in} \wedge s \in r'\} \cup \{s \mid s \in r \wedge s \in r'_{in}\} \cup \{s \mid s \in r_{on}r_{on} \wedge \exists s' \in r'_{in}r_{in} \vee s' \in r_{on}r_{in} \vee s' \in r'_{in}r_{on}\} \cup \{ss''s''' \mid ss''s''' \in r_{on}r_{on} \wedge ss''s''' \in MinUnit(rr')\}$
$r \text{ Minus } r'$	\equiv	$\{s \mid s \in r \wedge s \notin r'\} \cup \{s \mid s \in r_{in} \wedge s \in r'_{on}\} \cup \{s \mid s \in r_{on}r_{on} \wedge s \notin MinUnit(r_{on}r'_{on}) \wedge \exists s' \in r\} \cup \{s \mid s \in r_{on}r_{on} \wedge s' s'' \in r_{on}r_{on} \wedge ss''s''' \in MinUnit(rr') \wedge \exists s''s''' \mid s''s''' \in r \wedge ss''s''' \cap MinUnit(rr') = \phi\}$
$Contour(r)$	\equiv	$\{s \mid s \in r_{on} \wedge s \notin r_{ion}\}$
$Vertices(r)$	\equiv	$\{s \mid s \in r_{on}\}$
$r \text{ CommonBorder } r'$	\equiv	$\{s, s' \mid s, s' \in r_{on} \wedge s, s' \in r'_{on} \wedge \widetilde{ss'}\}$

TABLE II
SPATIAL-VALUED OPERATIONS ON SEG REGIONS

location) and computes the minimum distance neighbour in each sector. The CBN then computes whether it is part of any CLUT. If it is not and if it has at least one neighbour that belongs to r only amongst the closest neighbours, it sets the operation state to **true**. If the CBN is part of one or more CLUTs and if it has at least one neighbour that belongs to r only, and the segment between the CBN and that neighbour does not intersect any of the CLUTs, the CBN sets its state to **true**, otherwise **false**. A node with a **true** operation state that is a CBN or belongs to the interior of r and the boundary of r' , or to the boundary of r only, sets its edge state to **true**, otherwise **false**.

Result Collection Nodes that participate in the derived geometry add information about the latter to their *GIT*, if it does not exist already. If it does, they update the *TTL* for the geometry. They set the *GID* to the value received as part of the task message. For spatial-valued operations, type inference (i.e., reasoning from the input types) suffices to derive the result data type. For example, `plus` on geometries of type **regions** yields a value of type **regions**. Nodes set the *BoundaryNode* attribute as to the computed edge state. The *TTL* for the derived geometry is computed as the minimum of the *TTLs* of the input operands in case of binary spatial-valued operations, and to the input operand for unary ones.

VII. EXPERIMENTAL EVALUATION

This section presents experimental evidence that the algorithms scale well in terms of message complexity, energy consumption and response time in simulated, but realistic, deployment scenarios. At the time of writing, 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 the detection of induced geometries is run over the whole network. The cost of induced geometry detection is not reflected in experimental costs.

Experimental Set-Up We have implemented all the algebraic operations as distributed algorithms in nesC/TinyOS. For performance evaluation, we use PowerTOSSIM [18], a power modelling extension to TOSSIM that provides a per-node estimation of power. The specification of the nodes simulated is [Type = Mica2, Radio = CC1000, Energy Stock =

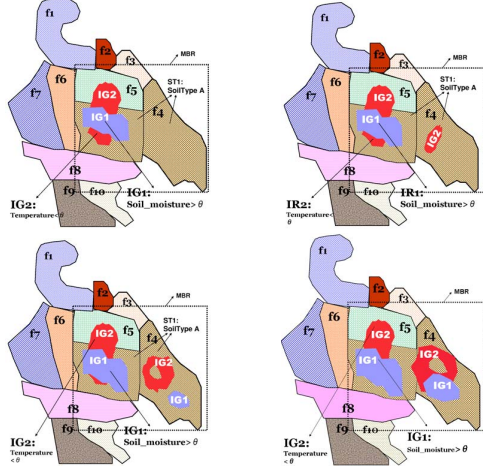


Fig. 1. Four Stages of an Evolving Phenomenon

31,320,000 mJ (2 Lithium AA batteries)]. We used TinyViz for modelling the behaviour of network. The maximum cardinality of the one-hop neighbourhood of a node is set to eight in all experiments.

Experiment 1: Behaviour wrt Dynamic Evolution This experiment explores the behaviour of the implemented algorithms when, given a network (consisting of 223 nodes), a transient physical phenomenon evolves through four stages. In terms of our motivating example, water infiltration and temperature influence soil moisture levels differently depending on the type of soil. Wine makers achieve greater control over the product by defining sub-blocks within the vineyard. Batch selection can then be decided based upon spatial locality, the type of soil and other factors. The four pictures (used in the four evaluations) in Fig. 1 show two induced geometries undergoing three change events. In these experiments f_5 comprises 63 nodes (out of which 39 nodes represent f_5 with soil type SA); and f_4 , 40 nodes. The hole in stages s_3 and s_4 comprises 7 nodes. The size of the remaining geometries is shown in Table III, where *Eval01* represents the first evaluation, and *IG1-1* denotes the first element of *IG1*. The task used in this experiment is:

((f_5 Intersection SA) Plus f_4) Minus (*IG1* Intersection *IG2*)

The results obtained are depicted in Fig. 2. The results are broken down into three components, viz., TD for task dissemination, DME for distributed membership evaluation and RP for result processing. It can be seen that, as expected, the TD phase is responsible for the majority of communication events. Our implementation exhibits slow rates of growth in terms of messages transmitted and that the energy consumed and the response time remain stable as the geometries that characterize an evolving phenomenon change over time. For the snapshot in *Eval04*, the amount of energy consumed by CPU and radio together is close to 77 kmJ (i.e., $\sim 0.25\%$ of the total energy stock). This is low enough to postulate that adding the energy required to induce the event geometries (not counted in Fig. 2(b)) is unlikely to significantly detract from the force of these conclusions.

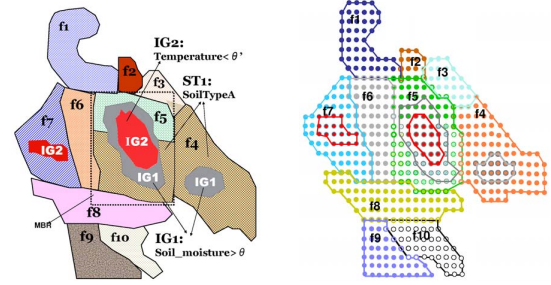


Fig. 3. (a) Intersecting Induced Regions (b) Example WSN over (a)

Experiment 2: Behaviour wrt Network Growth Here we assume that one wants to retrieve the MBR of an area where spraying is required. Fig. 3 shows field f_5 and two induced geometries. Table IV gives the size of geometries. The task used in this experiment is:

MinimumBoundingRectangle((f_5 Intersection *IG1*)
Intersection *IG2*)

After the derivation of the relevant geometry, its MBR is computed. This requires the construction of an aggregation tree [14] over location information. The construction involves two phases. In the *SEG leader election phase*, the nodes broadcast their ID and derived geometry ID. Upon receiving this information, each node in the derived geometry computes whether there is a neighbouring node with a smaller ID than its own. If there is, it assumes a non-leader role and sets a timer to wait for information from an elected leader node. If the wait time expires and it does not receive that information, it sends a tree construction message (TCM). If, instead, it has the smallest ID among its neighbours, it starts the tree creation process by broadcasting a TCM. A derived geometry node that has not already received the TCM, records this information, increments by one the *level* (i.e., *numberofhops*) attribute, set the *sourceID* attribute to its own ID and broadcasts the message. If, later, a node receives a TCM with better *leader* information (i.e., either a smaller distance the gateway or a smaller *leaderID*) than it records information, increments by one the *level* (i.e., *numberofhops*) attribute, set the *sourceID* attribute to its own ID and broadcasts the message. If a leader node receives the message with better leader information, it records this information and changes its state to non-leader. Upon receiving the TCM, a node waits for a predetermined time. Upon the expiry of the wait period, the node registers itself with its parent. This allows each node to learn how many children it has. A node with zero children recognizes itself as a leaf node.

After the tree has been set up, aggregation starts at the leaves. Each parent node depending on its *level* and the number of its children waits for predetermined time to receive information from its children with which to compute a local outcome. This is done by selecting the *minimum x-axis, y-axis* (i.e., bottom-left) location and *maximum x-axis, y-axis* (i.e., top-right) among its own location and the information coming from its children. It then forwards the local outcome to its parent. Finally, the SEG leader is responsible for transmitting the result towards *first-level* leader, using the routing

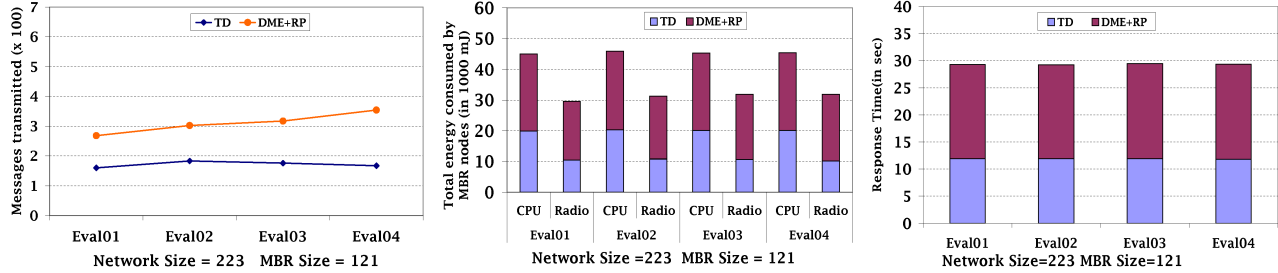


Fig. 2. Exp. 1: (a) Messages Transmitted (b) Energy Consumed And (c) Response Time, As The Geometry Evolves

	Eval1	Eval2	Eval3	Eval4
IG1-1	31	31	31	31
IG1-2	-	9	15	24
IG2-1	17	24	30	30
IG2-2	-	-	7	17

TABLE III
SIZE OF REGIONS IN EXP. 1

Nodes	f5	IR1 in f5	IR2 in f5	IR1 in f4	IR2 in f7
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. 2

Nodes	Nodes MBR IN	Bytes IN	Bytes (*100) OUT	Messages (#)		Energy (kJ)				Resp. (sec)	
				IN	OUT	IN		OUT		IN	OUT
						CPU	Radio	CPU	Radio		
166	48	32	73	242	1175	29	27	245	177	43	140
223	71	43	121	301	1902	40	37	472	357	46	197
299	106	62	157	430	2515	72	63	1032	646	50	249
400	120	79	237	598	3758	76	67	1496	835	55	306

TABLE V
IN-NETWORK (EXP. 2) V. OUT-OF-NETWORK (EXP. 3) PROCESSING

tree constructed during the task dissemination. The *first-level* leader then forwards the result towards the gateway. The results obtained are depicted in Table V under the IN columns (i.e., denoting in-network processing of the task). The energy consumption cost adds the energy consumed by every node in the MBR spent on task dissemination, distributed membership evaluation, tree construction, aggregation and result processing. The bytes and messages transmitted costs and the response time is the total upon the message reaching the gateway.

Experiment 3: Out-of-Network Processing The purpose of this experiment was to measure the cost in terms of bytes and messages transmitted, energy consumed and response time involved in sending all the event information from boundary nodes back to the gateway for processing outside the network. It aims to quantify the extent to which in-network processing offers savings wrt out-of-network processing. The experiment was run over the scenario in Fig. 3. In this experiment, it was 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. The results obtained are depicted in Table V under the column heading of *Out* (i.e., denoting out-of-network processing of the task on the received boundary information). Table V shows how, in the case of both in- and out-of-network processing, the bytes and messages transmitted, the energy consumed and the response time grow much faster as the network size grows than in the in-network case. It can be seen that all costs (i.e., bit and message complexity, energy consumption and response time) for out-of-network processing are very high compared to our in-network implementation.

VIII. CONCLUSION

This paper has described a framework for representing spatial values that model asserted, induced, and derived geometries in WSNs. It has described algebraic operations for the derivation of geometries inspired by the ROSE algebra and illustrated the techniques used for their implementation. The paper has shown, through empirical analysis, that in-network evaluation of spatial analytic tasks results in substantial savings in terms of energy consumption and response time compared to sending

boundary information of transient phenomena back to the gateway.

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*, Chapter on Barycentric Coordinates, pp. 216–221. Wiley, 1969.
- [4] C. Farah, C. Zhong, M. F. Worboys, and S. Nittel. Detecting topological change using a wireless sensor network. In *GI Science*, pp. 55–69, 2008.
- [5] I. Galpin, C. Y. A. Brenninkmeijer, F. Jabeen, A. A. A. Fernandes, and N. W. Paton. Comprehensive optimization of declarative sensor network queries. In *SSDBM*, pp. 339–360, 2009.
- [6] T. Griffiths, A. A. A. Fernandes, N. W. Paton, K. T. Mason, B. Huang, M. F. Worboys, C. Johnson, and J. G. Stell. Tripod: A comprehensive system for the management of spatial and aspatial historical objects. In *ACM GIS*, pp. 118–123, 2001.
- [7] J. K. Hart and K. Martinez. Environmental Sensor Networks: A revolution in the earth system science? *Earth-Science Reviews*, 78:177–191, 2006.
- [8] F. Jabeen and A. A. A. Fernandes. Impact on accuracy of deployment trade-offs in localized sensor network event detection. In *LOCALGOS*, 2008.
- [9] F. Jabeen and A. A. A. Fernandes. Distributed Spatial Analysis in Wireless Sensor Networks. *Submitted*, 2010.
- [10] J. Jiang and M. F. Worboys. Detecting basic topological changes in sensor networks by local aggregation. In *ACM GIS*, page 4, 2008.
- [11] B. Karp and H. T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *MOBICOM*, pp. 243–254, 2000.
- [12] 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.
- [13] 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.
- [14] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TAG: A Tiny AGgregation Service for Ad-Hoc Sensor Networks. In *OSDI*, 2002.
- [15] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TinyDB: an acquisition query processing system for sensor networks. *ACM TODS*, 30(1):122–173, 2005.
- [16] L. Mottola and G. P. Picco. Logical neighborhoods: A programming abstraction for wireless sensor networks. In *DCOSS*, pp. 150–168, 2006.
- [17] M. Schneider. *Spatial Data Types for Database Systems: Finite Resolution Geometry for Geographic Information Systems*. LNCS 1288. Springer, 1997.
- [18] V. Shnayder, M. Hempstead, B. Chen, G. W. Allen, and M. Welsh. Simulating the power consumption of large-scale sensor network applications. In *SensSys*, pp. 188–200, 2004.
- [19] T. Ulrich. Wireless network monitors h2o: System saves resources, increases yield in cabernet vineyard. *Wines and Vines Magazine*, July 2008.
- [20] M. Welsh. Exposing resource tradeoffs in region-based communication abstractions for sensor networks. *Computer Communication Review*, 34(1):119–124, 2004.
- [21] M. F. Worboys and M. Duckham. Monitoring qualitative spatiotemporal change for geosensor networks. *IJGIS*, 20(10):1087–1108, 2006.
- [22] Budhaditya Deb and Sudeept Bhatnagar and Badri Nath. STREAM: Sensor Topology Retrieval at Multiple Resolutions. *Kluwer Journal of Telecommunications Systems*, vol 26, pp. 285–320, 2003.
- [23] K. Chintalapudi and R. Govindan. Localized edge detection in sensor fields. *Ad Hoc Networks*, 1(2-3):273–291, 2003.