Decentralized Detection of Topological Events in Evolving Spatial Regions

Muhammad Jafar Sadeq[1], Matt Duckham[1] and Mike Worboys[2]

[1]*Department of Infrastructure Engineering, University of Melbourne, VIC 3010, Australia*
[2]*School of Computing and Information Science, University of Maine, ME*
Email: mduckham@unimelb.edu.au

Qualitative information about topological events, like the merging or splitting of spatial regions, has many important applications in environmental monitoring. Examples of such applications include detecting the emergence of "hot spots" in sea temperature around a coral reef; or the break up and dispersion of an environmental pollution spill. This paper develops and tests an efficient, decentralized spatial algorithm capable of detecting high-level topological events occurring to spatial regions monitored by a wireless sensor network. The algorithm, called INQUIRE, is decentralized because at no point does any single system element possess global knowledge of the entire system state. Instead, INQUIRE relies purely on a sensor node's local knowledge of its own state and the state of its immediate network neighbors. Experimental evaluation of the INQUIRE algorithm demonstrates that our decentralized approach can substantially improve scalability of communication when compared with efficient centralized alternatives.

Keywords: environmental monitoring, areal object, decentralized spatial computing, wireless sensor network, topological change

Received 15 December 2011; revised 3 March 2012

1. INTRODUCTION

Wireless sensor networks (WSNs) are increasingly important for geospatial applications like environmental monitoring. Early applications of WSNs to environmental monitoring generally adopted a "sense-and-transmit" approach, forwarding each node's sensor readings to a centralized computing system for subsequent event detection [1–3]. By contrast, recent research is increasingly focusing on *decentralized* algorithms for event detection.

In a decentralized system, no single system component possesses global knowledge of the entire system state [4]. As a consequence, decentralized algorithms can help to reduce the need for communication, making more scalable use of limited energy reserves in resource-constrained WSNs [5]. Decentralization effectively reflects *spatial* constraints to the movement of information in a WSN [6]: the greater the distance over which information is communicated, the more system resources are consumed. By defining scalable decentralized algorithms, nearby sensor nodes can efficiently collaborate in the network to perform partial or complete processing of sensor data (e.g., [7, 8]).

This paper presents a new decentralized algorithm for in-network detection of the topological events that occur during the evolution of dynamic geographic regions. These events include (dis)appearance, merging, and splitting of regions, as well as other events discussed in Section 3.1. The algorithm—In-Network Qualitative Identification of Region Evolution (INQUIRE)—can identify these fundamental topological events using only local (short-range) communication between neighboring nodes and with no centralized control. The evaluation shows how, and under what conditions INQUIRE can out-perform conventional "sense-and-transmit" distributed strategies, especially with respect to network energy resources.

Potentially, INQUIRE has applications to any WSN tasked with monitoring the evolution of environmental regions, including monitoring contours in dynamic spatial fields. For example, when monitoring an environmental pollutant spill, like an oil slick, it may be important to know if and when the oil slick breaks up (i.e., splits); tracking the appearance and evolution of "hot spots" in sea temperature can assist with managing the impacts to coral reef

health; and detecting topological changes in wildlife habitat connectivity may help in protecting sensitive conservation areas.

2. BACKGROUND

The INQUIRE algorithm detects ongoing qualitative topological changes to a monitored region, as they occur. INQUIRE is decentralized. As already noted, a decentralized system is a special case of a distributed system where no single system component possesses knowledge of the global system state [4]. In spatial computing, the need for decentralization arises from the spatial constraints to the movement of information [6]. Consequently, decentralized spatial algorithms are structurally distinct from many other established distributed systems, like neural networks (e.g., [9–11]), where constraints to the movement of information (if any) are not spatial.

Acknowledging these spatial constraints, INQUIRE restricts communication to region boundaries, which in turn promises greater scalability than involving the entire region (as in, for example, [12–14]). The efficiency improvements are expected because a relatively small proportion of nodes in the region are expected to be located at the boundary, and because boundary nodes are expected to be located close to other boundary nodes.

In addition to centralized algorithms for identifying nodes at region boundaries (e.g., active contours in image processing [15]), decentralized algorithms have been proposed in the literature. In NED [16], a node compares the average of the readings of its neighborhood to a threshold in order to determine whether it is on the boundary. A classifier approach is adopted in [17], where nodes estimate the location of a best-fit line to separate nodes into two sides of a boundary. A node's distance from that line is used to determine whether it is a boundary node.

By contrast, INQUIRE does not attempt to interpolate the location of the boundary. Neither do we aim to estimate or predict future states, in contrast to approaches like Markov random fields [18]; nor to optimize over a range of competing alternatives, such as by using particle filters [19]. Instead, INQUIRE adapts to the available level of detail, recruiting into the computation only those nodes in the neighborhood of the actual region boundary.

This approach is also used in the contour mapping algorithm, GBD (gradient boundary detection). In GBD, to approximate the contours, an ordered list of nodes outside a region's boundary are collected before being relayed to the sink [20]. By contrast, INQUIRE is instead concerned with in-network tracking of region/contour evolution. It therefore involves detection of a region's boundary and maintaining information along that boundary, and reports topological events instead of reporting snapshots of boundaries.

The contributions of this paper fill an important gap in our previous work. The INQUIRE algorithm efficiently monitors the spatial events that occur to evolving spatial regions. Previous work has described formally the evolution of spatial regions [21], leading to the definition and testing of an algorithm for an exhaustive set of topological events (merge, split, self-merge, self-split, appearance, and disappearance) in a monitored region in [22]. However, both these approaches require the communication network be structured as a triangulation (and in the case of [22] as the Delaunay triangulation). Another approach to monitoring region evolution that does not require a maximal planar communication network is presented in [23]; however, this work does not restrict communication to the boundary, and instead requires coordination between *all* nodes inside the monitored region. By contrast, the INQUIRE algorithm is assumes only minimal geometric information, in the form of the cyclic ordering of neighbor nodes; and operates purely at the boundary of the monitored region. An existing static algorithm for querying a snapshot of a region's topological structure [24] can be used for one-off initialization or periodic error-checking for INQUIRE, discussed further in Section 7.3. In cases where coordinate information is available to nodes, an existing decentralized area computation algorithm described in [25] is one efficient option for distinguishing two possibilities during a region self-merge, discussed in Section 5.4. Finally, although our algorithm does not require any particular network structure, the communication neighborhood does structure the decentralized computation, and so different neighborhood structures (e.g., UDG versus planar graph) can impact the topological events that can be detected, explored in [26].

3. FORMAL FOUNDATIONS

This section synthesizes a number of existing formal models, providing precise definitions of sensor networks, the dynamic spatial regions of interest, and their interrelationships.

3.1. Areal object model

In this paper, we are concerned specifically with tracking the evolution of geographic *areal objects*. Although we have used the term "(spatial) region" loosely up to this point, a *region* is conventionally taken to refer to a connected area within the plane (more strictly: a connected, bounded, regular closed subset of the plane). Regions may or may not have holes. An *areal object* is the finite union of a collection of disconnected regions. In general, an areal object may have holes in which there are islands, holes within islands, and so forth. (Also note a duality here, in that the holes may be thought of as regions containing elements not in the areal object.)

These areal objects may be truly binary (for example, an oil-slick defined using the boundary between locations that contain oil-covered and oil-free water) or may be derived from thresholding of continuous fields (for example, a temperature "hot-spot" defined using the boundary between locations that exhibit temperatures above or below, say, 24°C). The areal objects in this work are assumed to have crisp boundaries.

To represent dynamic areal objects we begin by adopting a standard generic model from [27, 28]. We can define an *inside* relation on the collection of regions and holes, in a topologically intuitive way. Thus, in the example shown in Figure 1, region 3 is inside hole 2, which is inside region 1, itself inside the exterior 0. After [27, 28], a tree can be used to model the inside relation between regions and holes. Each vertex in the tree represents a region or a hole, and a vertex's direct descendants in the tree are the holes or regions directly inside it. Figure 1 illustrates this representation.

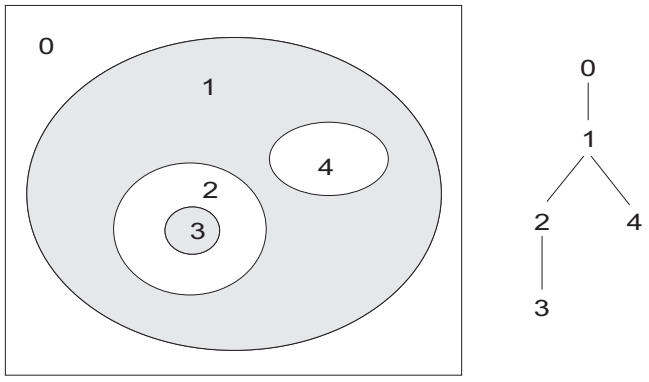

FIGURE 1. Tree representation of an areal object (after [27, 28])

It should be noted that alternative topological models of areal objects do exist, such as [29, 30]. These models have the advantage of being able to describe a wider range of topological relations (in particular, cases where region components have disjoint interiors but overlapping boundaries, excluded from our model). However, these more expressive models come at the cost of increased complexity, and so we leave the investigation of these alternatives as a topic for future work.

Building on the model of areal objects presented in [28], [31] have shown, using tree morphisms, that there are just six fundamental topological events (shown in Figure 2):

- the *appearance* or *disappearance* of a hole or region component, requiring the insertion of a new leaf node into the tree or deletion of an existing leaf node from the tree, respectively;
- a *merge* of disconnected holes or region components, requiring the merging of two existing nodes at the same level of the tree
- a *split* into disconnected holes or region components, requiring the splitting of an existing node into two at the same tree level;
- a *self-merge* of a hole or region component, requiring the merging of two nodes at different levels in the tree (destroying an intermediate contained hole or region component); and
- a *partial-split* of a hole or region component, splitting a node into two at different levels (engulfing part of the containing hole or region component).

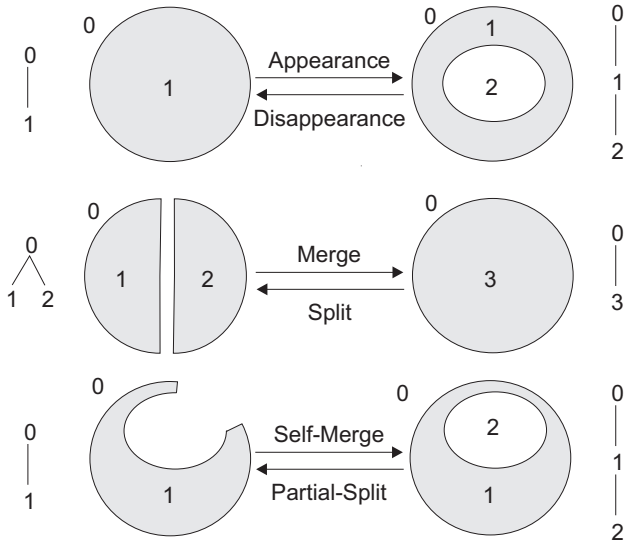

FIGURE 2. Six fundamental topological events for a dynamic areal object, after [31]

In [31], these six changes are proven *exhaustive*, and so constitute the complete set of topological changes. Building on their model, the INQUIRE algorithm performs in-network identification of these events occurring to a monitored areal object.

3.2. Sensor network model

This paper adopts a conventional model of sensor networks as a *communication graph* G, where vertices in the graph represent sensor nodes, and edges in the graph represent direct radio communication links between neighboring nodes.

DEFINITION 3.1. *The* communication graph *is an undirected, unweighted graph* $G = \langle V, E \rangle$, *where* V *represents the set of all sensor nodes, and each edge* $\{s, s'\} \in E$ *represents a direct (one-hop) communication link between nodes* $s, s' \in V$. *For a node* s, *the collection of its immediate neighbors in the communication graph* $\{s' \in V | \{s, s'\} \in E\}$ *is denoted* $nbr(s)$.

In the most basic case, the communication graph G may represent the physical wireless network

connections. These physical connections are often modeled using the unit disk graph (UDG), where there exists a link between any two nodes closer than some maximum communication range c (i.e., $\{s, s'\} \in E$ if $|ss'| \leq c$) [32]. However, G can also be generated by more sophisticated *overlay network* structures, such as a triangulation, the relative neighborhood graph (RNG), or the Gabriel graph (GG) [33].

In addition to the purely topological information contained within the communication graph G, distinguishing certain types of events additionally requires geometric information about the cyclic ordering of neighbors around a node. The cyclic ordering is be represented formally as the function d:

Definition 3.2. *The cyclic ordering of neighbors around a node is modeled as a function $d : V \times V \to V$, such that $d(v_1, v_2) = v_3$ where $v_2, v_3 \in nbr(v_1)$, and v_3 is the next node in the neighborhood of v_1 counterclockwise after v_2.*

Figure 3 illustrates the cyclic ordering of neighbors around a node, where $d(v_1, v_2) = v_3$, $d(v_1, v_3) = v_4$, and so on until $d(v_1, v_7) = v_2$.

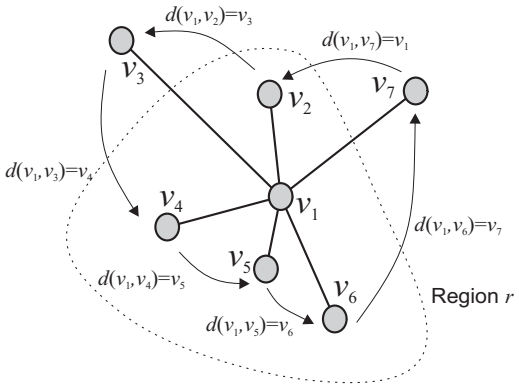

FIGURE 3. Example of cyclic ordering of neighbors around a node

The question remains: how should the WSN generate this cyclic ordering information? Many different techniques for node localization in WSN have been proposed and used. Nodes equipped with (virtual or geodetic) coordinate positioning capabilities can derive the cyclic ordering in a straightforward way using the geometry of neighbor positions. However, even in cases where coordinate position is unavailable, the cyclic ordering might also be generated from other sources, such as direction-finding techniques. Irrespective of the specific localization system used, INQUIRE is expected to be relatively robust to errors in quantitative coordinates or directions, because it relies only on imprecise, *qualitative* spatial information about the relative cyclic ordering.

3.3. Relationship between areal object and sensor network models

In linking together the models of the areal object and the sensor network, we require three structures:

1. a function n representing the information each node can locally sense about the areal object;
2. a partition of the communication graph into connected subgraphs, each of which contains nodes that sense part of the same hole or region; and
3. a relation over the partition of connected subgraphs, which reflects the containment between holes and region components.

All these structures are time-varying over some totally ordered set of discrete times T. First, to represent the information a node can locally sense about the areal object:

Definition 3.3. *The function $n : V \times T \to \{0, 1\}$ represents whether a sensor node s senses it is in, $n(s, t) = 1$, or out, $n(s, t) = 0$ of the areal object at time $t \in T$.*

Next, the values sensed by nodes can induce a partition the communication graph of the network into (hole and region) components. Formally:

Definition 3.4. *A* piece *of the communication graph $G = (V, E)$ at time t is a set of nodes $V' \subseteq V$ such that for any $s' \in V'$ and $s \in nbr(s')$ then $n(s', t) = n(s, t)$ if and only if $s \in V'$. The subgraph $G' \subseteq G$ induced by a piece V' is called a* component *if G' is connected.*

At any time t, the set of all components $\mathcal{G}(t)$ by definition forms a partition of G. We designate one particular component $X \in \mathcal{G}(t)$ as the *exterior* component. This exterior component will form the root of the containment tree for the areal object, and reflects our assumption that the sensor network extents are large enough to cover the areal object being monitored. A containment tree is then formed by using the set of all components as nodes of the tree, and an irreflexive relation to indicate immediate containment between components, in the following way:

Definition 3.5. *Two distinct components $C, C' \in \mathcal{G}(t)$ are said to be* c-adjacent *at time t, $C \lhd_t C'$, if and only if: a) the subgraph of G induced by the vertices of C and C' is connected (i.e., C and C' are adjacent in G); and b) there exists no path in G from any node of C to the exterior component $X \in \mathcal{G}(t)$ that does not pass through C'.*

We make the further constraints that 1. G is connected; 2. the components form a tree with root X under c-adjacency (i.e., for any component $C \in \mathcal{G}(t)$ where $C \neq X$, there exists a *unique* $C' \in \mathcal{G}(t)$ such that $C \lhd_t C'$); and 3. that all non-leaf components contain more than one node from the communication graph G (i.e., there exist no $C, C' \in \mathcal{G}(t)$ such that $C' \lhd_t C$ and

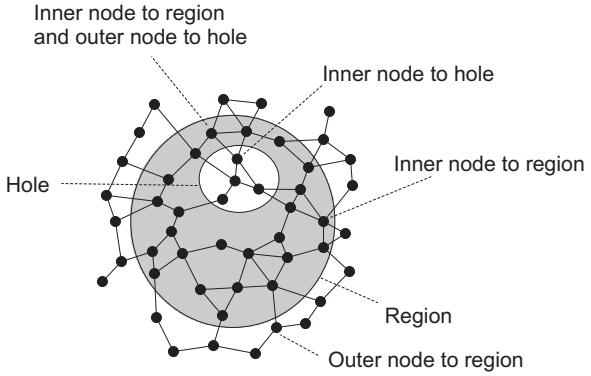

FIGURE 4. Boundary node classification for an areal object with one hole

C contains only one vertex).

Finally, we can now define the sets of nodes at the inner and/or outer boundaries of some hole or region component.

DEFINITION 3.6. *A node s is considered an* inner *boundary node at time t if there exists $s' \in nbr(s)$ such that $s \in C$ and $s' \in C'$, where $C \lhd_t C'$. Correspondingly, an* outer *boundary node $s \in C$ at time t has $s' \in nbr(s)$ such that $s' \in C'$, where $C' \lhd_t C$. The set of all inner boundary nodes at any time t is denoted $In(t)$, while the set of outer boundary nodes is denoted $Out(t)$. $Bdy(t) = In(t) \cup Out(t)$ is the set of all boundary nodes in the WSN.*

Note that a node may be an inner boundary, outer boundary, both, or neither, as illustrated in Figure 4. In Figure 4 the black dots represent sensor nodes and edges represent communication links between nodes, together representing the communication graph G.

3.4. Auxiliary structures

It is convenient to define two further structures to assist with concise discussions of the node states and transitions in later sections.

First, from the perspective of each individual node $s \in V$, looking around the cyclic ordering of the neighbors of s at a particular time $t \in T$, enables the neighbors to be partitioned into contiguous "blocks" that sense region components or holes. More specifically:

DEFINITION 3.7. *The relation $R_{s,t}$ is defined on $nbr(s)$ such that $s_1 R_{s,t} s_2$ if and only if $d(s, s_1) = s_2$, and $n(s_1, t) = n(s_2, t)$. The reflexive, symmetric, transitive closure of $R_{s,t}$ is written $\sim_{s,t}$. Note that $\sim_{s,t}$ is by definition an equivalence relation.*

Intuitively, neighbors of s are related by $\sim_{s,t}$ if they are all in or out of the areal objects at time t and together form a continuous block in the cyclic ordering around s. For example, with reference to Figure 3 and the region r, the neighborhood of v_1 can be partitioned into $nbr(v_1)/\sim_{v_1,t} = \{\{v_2\}, \{v_3\}, \{v_4, v_5, v_6\}, \{v_7\}\}$.

Second, in any given time step, those sensor nodes that have detected a change in the environment are termed "activating," while those that have not are termed "passive." Activating nodes fulfill an important role in the INQUIRE algorithm, formally defined as follows:

DEFINITION 3.8. *For a time interval $[t, t']$, nodes that have detected a change in sensor reading are said to be* activating. *Those nodes that have not detected a change we term* passive. *The set $Activate(t, t') \subseteq V$ is defined as the set $\{s \in V | n(s, t) \neq n(s, t')\}$. The set $Passive(t, t') \subseteq V$ is defined as the set $\{s \in V | s \notin Activate(t, t')\}$.*

In this paper we are primarily interested in nodes that activate over consecutive time steps (i.e., $Activate(t, t')$ where t is the immediate predecessor of t' in T, $t \prec t'$).

4. SENSOR NODE AUTOMATON

Building on the formal model set out in the previous section, this section constructs a computational model of decentralized, in-network detection of topological change. Sensor nodes are modeled as automata using their different boundary states (in Definition 3.6) and the possible transitions between boundary states. By using information about the local boundary state, we can determine whether some larger-scale topological change is occurring without requiring global communication between nodes (i.e., using only local, p2p communication).

At this point, however, a key simplifying assumption is that the sensor network's sensing rate is high enough to detect *incremental* changes in the underlying dynamic field (as in [21, 22]). In other words, for any two consecutive time steps, $t, t' \in T$ where $t \prec t'$, there is only ever one activating node (i.e., $t \prec t'$ implies $|Activate(t, t')| \leq 1$). While this is conceptually straightforward, in practice this assumption is limiting. A key focus for ongoing research is relaxing this simplifying assumption (see Section 7.1).

4.1. Node states and transitions

Each sensor node s has four states depending on whether $s \in In(t)$ and $s \in Out(t)$, shown in Table 1.

Given the restriction of incremental change, introduced above, not all state transitions are possible for a node that is activating. Table 2 enumerates the 16 different state transitions, highlighting with a tick those transitions that are possible for an activating node given incremental change. The row headings give the initial state of the node before it activates; the column headings list the state of the node after it is activated (although since the table is symmetrical the converse interpretation is also valid).

The following subsections provide examples of all the *possible* transitions. Intuitively, some transitions

TABLE 1. Four node boundary states

State	Formal description	Informal description
NB	$s \notin In(t)$ and $s \notin Out(t)$	s is not a boundary node
IB	$s \in In(t)$ and $s \notin Out(t)$	s is an inner boundary node only
OB	$s \notin In(t)$ and $s \in Out(t)$	s is an outer boundary node only
IOB	$s \in In(t)$ and $s \in Out(t)$	s is both an inner and outer boundary node

TABLE 2. Possible state transitions for activating node assuming incremental change

	NB	IB	OB	IOB
NB	✗	✓	✗	✗
IB	✓	✗	✓	✗
OB	✗	✓	✗	✓
IOB	✗	✗	✓	✗

are impossible because an activating node must "cross" a boundary (transition from sensing a region component/hole to sensing a hole/region component). Given incremental change, only certain transitions can be achieved in one step for such a node crossing the boundary. A proof of the non-existence of those transitions marked with a cross in Table 2 can be constructed as follows.

Proof. We take each case in turn for a unique activating node $a \in G$ at consecutive timesteps t, t'.
NB→NB: For an activating node a in state NB, an arbitrary neighbor $a' \in nbr(a)$ must sense the same value at time t, $n(a,t) = n(a',t)$. At time t', a's sensed value must have changed, $n(a,t) \neq n(a,t')$, but due to incremental change a' must continue to sense the same value, $n(a',t) = n(a',t')$. Therefore, $n(a,t') \neq n(a',t')$, and so a cannot be in state NB at time t'.
IB→IB: An activating IB node a in component $C \in \mathcal{G}(t)$ must have neighbors in the containing region $C' \in \mathcal{G}(t)$ and possibly also in C, i.e., $nbr(a) \subseteq C \cup C'$ where $C \lhd_t C'$. Further, a path from any $c' \in C - \{a\}$ to the exterior X must still pass through $C' \cup \{a\}$ at time t'. Consequently, $C \lhd_t C'$ implies it is not possible that $(C' \cup \{a\}) \lhd_{t'} (C - \{a\})$. Because node a must still have only neighbors in $C \cup C'$ at time t', it follows that a must not have transitioned to state IB.
NB→OB: As for NB→NB, at time t, node a must sense the same value—and so be in the same component—as all its neighbors. A constraint to Definition 3.5 above was that containing components may not be singleton nodes. Thus, at time t', a cannot be in state OB.
OB→OB, IOB→IOB, IB→IOB: These proofs follow the same structure as case IB→IB.
NB→IOB: This proof is as for NB→OB.
OB→NB, IOB→NB, IOB→IB: The transitions are symmetrical, so these proofs are as for their converses, NB→OB, NB→IOB, and IB→IOB respectively. □

A similar table analysis is not provided for passive nodes. This is because activating nodes are the ones that detect changes, and are the best suited to determine what event has occurred based on the change in their boundary state. Passive nodes only receive information from activating nodes and update their own states if necessary in order to maintain a correct state (for example, if a region expands, a passive node that was not adjacent to the region can become outer to the region).

The following three subsections describe the three pairs of transitions that were presented in Section 3.1.

4.2. Appearance/disappearance

For an activating non-boundary (NB) node, only one transition, to a inner boundary state, is possible. Figure 5 illustrates the situation for activating node A (square) initially in a NB state. Given that only A is permitted to become active in this time step (incremental change) and that all neighbors of A must initially sense the same region component/hole as A, no other transitions are possible. In fact, this transition corresponds to one of the six fundamental topological events of dynamic areal objects set out in Section 3.1: the *appearance* of a region component/hole. The reverse transition, from IB to NB is also possible and corresponds to the *disappearance* of a region component/hole. Note that the passive nodes (those in the neighborhood of the active node) also have a (different) state transition associated with appearance/disappearance (to/from an OB state).

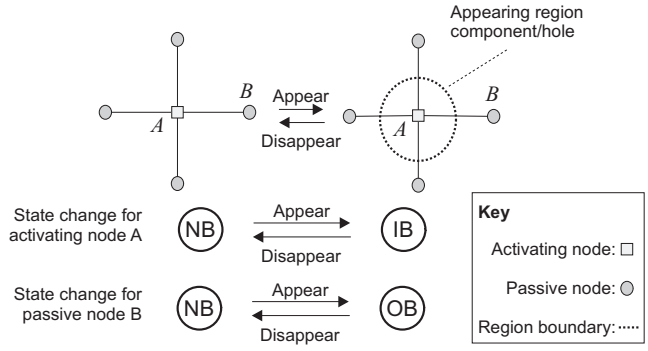

FIGURE 5. Transition #1: appearance/disappearance

4.3. Split/merge and expand/contract

For an activating inner boundary (IB) node, we have already seen one possible transition to a non-boundary (NB) state (Section 4.2). A second transition to an outer boundary (OB) state is also possible. Figure 6 gives an example of such a transition from IB to OB. The transition in Figure 6 corresponds to the topological split described in Section 3.4. The

corresponding reverse transition from OB to IB is also possible, and can correspond to a topological merge. Note also that the passive nodes can experience (two) different transitions associated with this change.

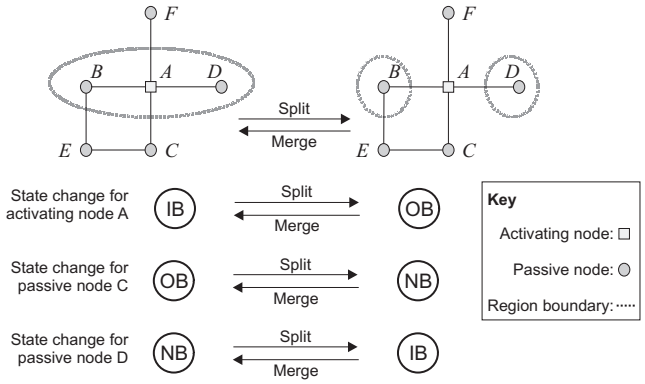

FIGURE 6. Transition #2a: merge/split

It is also possible for the same transitions between IB and OB to occur when no topological change has occurred. For example, Figure 7 shows the same state transitions for a non-topological change where the region component/hole simply expands or contracts (i.e., a new node is added to or removed from a region component/hole without any topological change to the underlying areal object graph). Thus, unlike transition #1 in Section 4.2, the state changes alone do not enable us to distinguish between topological merge/split and non-topological expansion/contraction. Instead, some additional processing will be necessary to identify the event that is occurring. This is discussed in Sections 5.3 and 5.4.

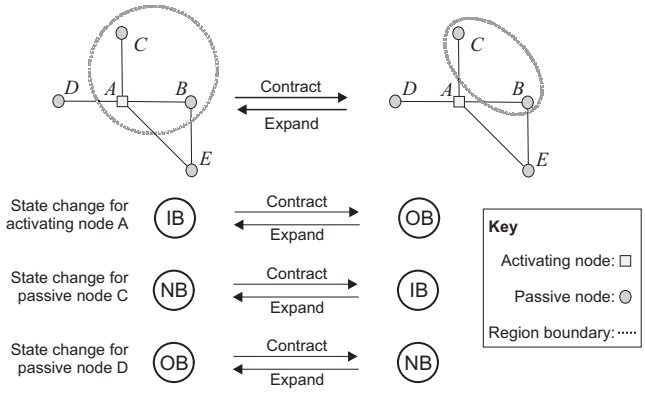

FIGURE 7. Transition #2b: expansion/contraction

4.4. Self-merge/partial split

For an activating outer boundary (OB) node, we have already seen one possible transition to an inner boundary (IB) state (Section 4.3). A second transition to an inner/outer boundary (IOB) state is also possible. For example, Figure 8 gives an example where the activating node transitions from OB to IOB. This transition corresponds to the topological self-merge described in Section 3.4. The reverse transition from IOB to OB corresponds to a topological partial-split.

Again, passive nodes may experience different state changes. Nodes B to H in Figure 8 are chosen to exhibit all the possible changes that may occur in passive nodes during a self-merge/partial split.

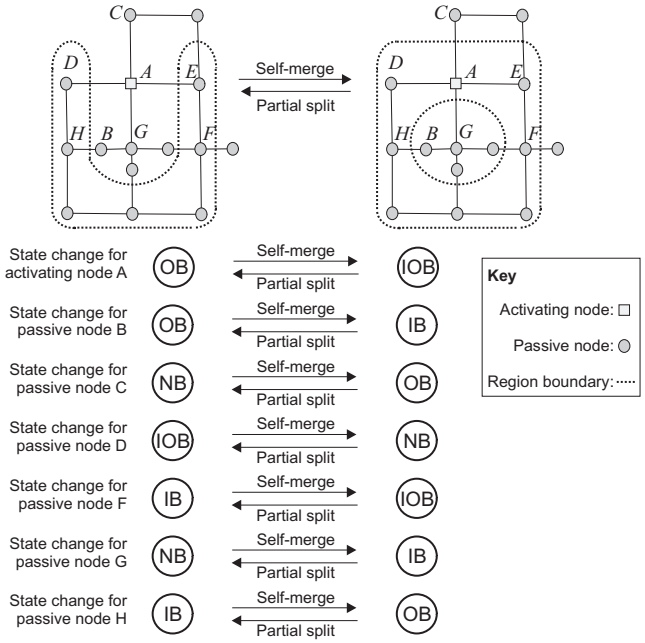

FIGURE 8. Transition #3: self-merge/partial-split

4.5. Summary

In summary, while there are many possible state transitions for passive nodes, there are only six possible transitions for activating nodes in our model. Four of these six state transitions correspond directly to the fundamental topological changes appearance, disappearance, partial split, and self merge. The two further state transitions correspond to both the two remaining fundamental topological changes, split and merge, but may also correspond to non-topological expansion or contraction of an areal object. Figure 9 summarizes these possible state changes for an active node.

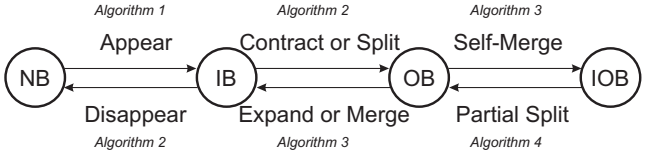

FIGURE 9. Possible state changes for an active node

The algorithm in the following section describes how each change in Figure 9 can be identified. Since this is a complete list of changes [31], the algorithm presented is

also complete (at least for incremental changes). Figure 9 also points to the part of the algorithm that handles each particular change.

5. INQUIRE DESCRIPTION

Up to this point this paper has presented a formal model of dynamic areal objects and a sensor network monitoring those areal objects (Section 3). Building on this formal model, the previous section (Section 4) indicated how a computational model of sensor nodes as automata can be used to locally detect topological changes using boundary state transitions. Building on both these models, this section presents INQUIRE, a decentralized algorithm for locally detecting large-scale topological changes by maintaining boundary state information for each sensor node. By performing a little extra communication in advance for boundary state maintenance, INQUIRE aims to reduce the overall communication required to detect topological events (addressed further in Section 6).

5.1. Algorithm preliminaries

When a node detects it has been activated (i.e., its sensor reading has changed), it must update its own boundary state information, as well as ensure the boundary states for passive nodes are updated. The specific action taken by an activating node depends on its initial state. In some cases, the topological event will be obvious from the initial state (i.e., changes from NB or IOB states, where only one state transition is possible). In other cases, the activating node needs to perform some processing to distinguish between different kinds of change. The activating and passive nodes need to analyze their neighborhood (perhaps by simply checking tables that they maintain) to determine their new states. INQUIRE in this section has two objectives: 1) to determine what topological event has occurred, if any, and 2) to determine the new state.

Without loss of generality, we assume each component in the containment tree can be associated with a unique identifier. The practical issue of ensuring all regions are assigned unique identifiers in the distributed sensor network is returned to in Section 7.4. Each sensor node s then maintains two *local* state variables that can change over time, $OutN(s,t)$ and $InN(s,t)$. $OutN(s,t)$ denotes the set of identifiers of components in the containment tree for which s is an outer boundary node at time t; similarly, $InN(s,t)$ denotes the set of identifiers of components in the containment tree for which s is an inner boundary node at time t. Because a node can be at the inner boundary of at most one region, $|InN(s,t)| \in \{0,1\}$.

The combination of these two variables maps directly to the four boundary states given in Table 1 (i.e., $s \in In(t) \leftrightarrow |InN(s,t)| = 1$, $s \in Out(t) \leftrightarrow |OutN(s,t)| \geq 1$). A node s can also compare its readings to its neighbors' to determine locally whether it is a boundary node at time t ($s \in Bdy(t)$). As long as there is one neighbor with a different reading ($\exists s' \in nbr(s)$ such that $n(s,t) \neq n(s',t)$) then s is a boundary node.

Finally, we further assume that the sensors are initialized with their correct state. There are a variety of ways that this initialization might be achieved, but we return to this question later (Section 7.3). The following algorithm describes the actions that are necessary to maintain each node's state as the dynamic areal object changes. For space concerns, the algorithm presented leaves out how passive nodes update their state. For ease of discussion, the algorithm is divided into four parts based on the activating node's current state.

5.2. Current state NB

From Figure 9, an activating node in the NB state knows that a region component/hole has appeared (since no other changes may occur from the NB state). In Algorithm 1 the activating node s selects an unused region identifier, r (line 2), updates its own state by storing this region identifier (line 3), and informs all of its neighbors that they are outer boundaries to the new region (line 4). We return to the practical issue of how a node selects an unused region identifier later (Section 7.4).

Algorithm 1: Current state NB

1 **Local variables**: activating node s; current time step t_i; previous time step t_{i-1}, where $t_{i-1} \prec t_i$
2 Select a previously unused unique component identifier r;
3 Set $InN(s,t) \leftarrow \{r\}$;
4 Broadcast r to $nbr(s)$;

5.3. Current state IB

From Figure 9, an activating node that transitions from state IB indicates that any one of three possible changes are occurring: disappearance, contraction, or split. Algorithm 2 details the processing necessary to distinguish these cases.

If the activating node s was the only node in the region, a change implies that the region no longer exists (since the last node in the region has left the region, Algorithm 2, line 3).

If there is at least one neighbor s' that is not an outer boundary to the component with identifier r, then s' is in component r. Therefore r has not disappeared. Either r is contracting or it has split. To distinguish these two possibilities, s sorts $nbr(s)$ according to direction, using the equivalence relation $\sim_{s,t}$ (Definition 3.7). The quotient set $nbr(s)/\sim_{s,t}$ will contain subsets of nodes in $nbr(s)$ that are in the same

"block" of the cyclic ordering. These "blocks" represent the alternating sections of region component and hole around s.

If there are fewer than four equivalence classes (blocks) in $nbr(s)/\sim_{s,t_i}$, then r has contracted (see Figure 7). In this case, s becomes outer to component r (Algorithm 2, line 6).

If there are four or more equivalence classes in $nbr(s)/\sim_{s,t_i}$, then r has split (Figure 6). In this case, r must be replaced by distinct labels for each new region component/hole formed (Algorithm 2, line 7). From each split region of r, a node is elected to select a new label and propagate it to all nodes in that region's boundary. The need to update the region label means that split (and its reverse, merge) can be detected locally, but the maintenance of the state is not local.

A combination of split and partial-split can occur (Figure 10). In this case, the split event is resolved before the partial-split event (Section 5.5).

Algorithm 2: Current state IB

```
1  Local variables: activating node s; current time
   step t_i; previous time step t_{i-1}, where t_{i-1} ≺ t_i;
2  r ← InN(s, t_{i-1});
3  InN(s, t_i) ← ∅;
4  if s ∈ Bdy(t) then
5  |  OutN(s, t_i) ← OutN(s, t_{i-1}) ∪ {r};
6  |  if |nbr(s)/∼_{s,t_i}| ≥ 4 then
7  |  |  foreach new region/hole do
8  |  |  |  Generate a new label r' and distribute it
   |  |  |  around the boundary;
9  |  |  end
10 |  end
11 end
```

We do not specify here a procedure for routing around the boundary. In cases where the communication graph is planar, a procedure akin to face-routing [34, 35] can be used for routing around the boundary (cf. [24]). However, recent theoretical advances, like the coordinate-free approach of [36], are yielding a variety of techniques that might be easily adapted to generate a cycle around the boundary of a region even where the network is not planar, and nodes have no information about their coordinate position (e.g., [37–39]).

5.4. Current state OB

From Figure 9, an activating node that transitions from state OB indicates that any one of three possible changes are occurring: expansion, merge, or self-merge. Algorithm 3 details the processing necessary for these cases.

If the activating node s is outer to more than one region, a merge has occurred (Figure 6). In this case, s selects at random a label r from $OutN(s, t_{i-1})$ to be the label of the new merged region and updates the entire boundary with this new identifier r (using one of the procedures outlined above).

If the activating node s is outer to only one region r then the change may be either a self-merge or expansion. In either case, s changes from being outer to r to become inner to r and informs its neighbors of the change.

If $|nbr(s)/\sim_{s,t_i}| \geq 4$, a self-merge has occurred (Figure 8). A hole that is formed from this self-merge needs to be assigned a new region label. The activating node s selects a node in that hole to generate the label and distribute it around the hole boundary.

Note that it is possible for a combined merge and self-merge, and the reverse, to occur (see Figure 10b). To determine when such an event occurs, observe from Figure 10a that when 3 regions merge, there are 6 adjacent sections around the active node A. In general, for a pure merge at activating node s, the number of adjacent equivalence classes is twice the number of regions that s is outer to ($|nbr(s)/\sim_{s,t_i}| = 2 * |OutN(s, t_i)|$). If there is a combination of merge and self-merge, as in Figure 10b, there are 6 adjacent sections but $|OutN(s, t_i)| = 2$. Therefore, as long as $OutN(s, t_i) \geq 2$ and $|nbr(s)/\sim_{s,t_i}| \geq 3 * Out(s, t_i)$ there is a simultaneous merge and self-merge. The procedure in this case is to first treat the event as a merge, then assign labels to the holes. In practice, with thousands of randomized simulations, this event never occurred. Therefore, although the implementation of INQUIRE includes this possibility, it is left out of the presented algorithm for simplicity.

Algorithm 3: Current state OB

```
1  Local variables: activating node s; current time
   step t_i; previous time step t_{i-1}, where t_{i-1} ≺ t_i;
2  if |OutN(s, t_{i-1})| > 1 then
3  |  Select random r ∈ OutN(s, t_{i-1});
4  |  OutN(s, t_{i-1}) ← ∅;
5  |  if s ∈ Bdy(t_i) then
6  |  |  InN(s, t_{i-1}) ← {r};
7  |  end
8  else
9  |  OutN(s, t_i) ← OutN(s, t_{i-1}) − {r};
10 |  InN(s, t_i) ← {r};
11 |  if |nbr(s)/∼_{s,t_i}| ≥ 4 then
12 |  |  A node in each hole generates new region
   |  |  label and distributes it;
13 |  end
14 end
```

An important problem facing INQUIRE is in determining on which side of a self-merge the hole lies. Figure 11 illustrates that deciding on which side of a self-merge the hole lies may require information about the region shape that extends beyond the immediate neighborhood

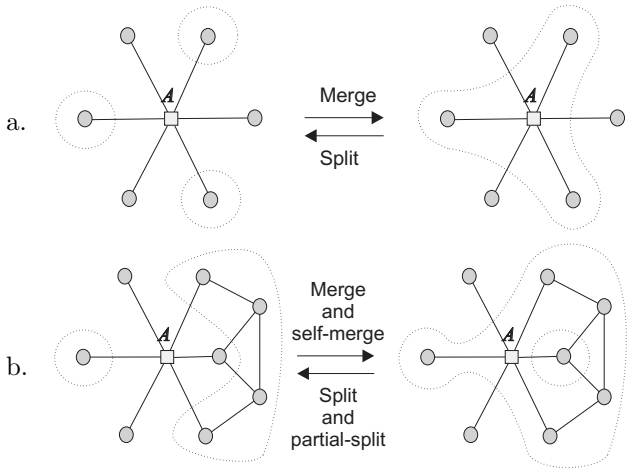

FIGURE 10. Simultaneous merge and self-merge

of the activating node A (assumed to have just detected a self-merge).

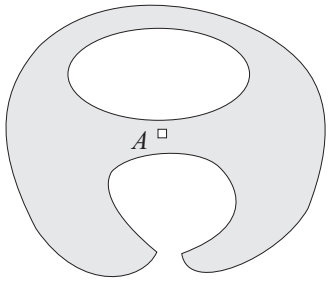

FIGURE 11. Where is the hole?

In cases where nodes possess information about their coordinate position (for example, via GPS), a decentralized algorithm for determining the area enclosed by each region/hole boundary can be used (such as provided in [12, 25]). In the case where one boundary contains another, like that in Figure 11, the boundary that encloses the larger area must contain the smaller boundary.

In cases where there is no coordinate information, the self-merge can be detected but the hole may be left unidentified. Consequently, changes at the hole's boundary may be misinterpreted as changes to the region's outer boundary. These misinterpretations are localized, only occurring at the boundary of this particular unidentified hole. Therefore, the misinterpretations are also temporary, lasting only as long as the hole remains in the areal object. Further heuristics might also reasonably be applied, such as estimating the relative size (and so containment) of the two boundaries by comparing their length in terms of the number of nodes in each boundary cycle.

5.5. Current state IOB

From Figure 9, an activating node initially in state IOB indicates that a partial-split has occurred. In Algorithm 4, the activating node s records the fact that it is no longer inner to region r, and has instead become outer to region r (lines 2–4). It then informs its neighbors of the change.

Algorithm 4: Current state IOB

1 **Local variables**: activating node s; current time step t_i; previous time step t_{i-1}, where $t_{i-1} \prec t_i$;
2 $r \leftarrow InN(s, t_{i-1})$;
3 $InN(s, t_i) \leftarrow \emptyset$;
4 $OutN(s, t_i) \leftarrow \{r\}$;
5 Inform $nbr(s)$ of partial-split in r;

6. EVALUATION

A sensor network simulation was built using Repast (`http://repast.sourceforge.net/`), and INQUIRE was implemented on the simulated nodes. Repast allows the number and location of nodes, the network dimensions, and the dynamic environmental field to be varied in different simulation runs without the need to deal with low-level networking issues. In the following experiments, nodes are located at random in the network.

For our evaluation, the Delaunay triangulation [40] was used to create the links between nodes. In practice, it is unlikely that the communication network of a WSN will form a Delaunay triangulation. Despite this, the Delaunay triangulation was chosen for the evaluation because it more faithfully preserves spatial proximity than other network structures, shown to help in reliably detecting qualitative spatial events [26]. However, a range of other planar and non-planar network structures have also been tested.

An example of a network, with 500 nodes, a field size of 500×500 square units, and communication range 50 units, is shown in Figure 12. The black area in the figure represents the state (location) of the areal object that the WSN is tracking at that particular point in time.

Each node was equipped with INQUIRE. Thus INQUIRE runs in parallel on all nodes at the same time with no centralized control. A series of experiments was conducted on arbitrary sequences of topological changes to the underlying field to investigate the relationship between the level of communication and number and types of topological events.

In the simulations, all eight (six topological and two metric) spatial changes were correctly identified by the WSN, as long as network density was comparable to or finer than the level of granularity of the underlying region.

As discussed in the introduction, WSNs are highly resource-constrained computing environments, with energy being the overriding resource constraint, and wireless communication being the most energy intensive

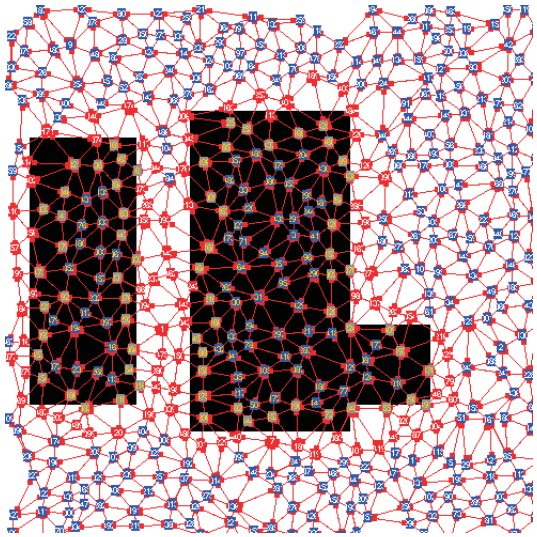

FIGURE 12. A sample simulated WSN

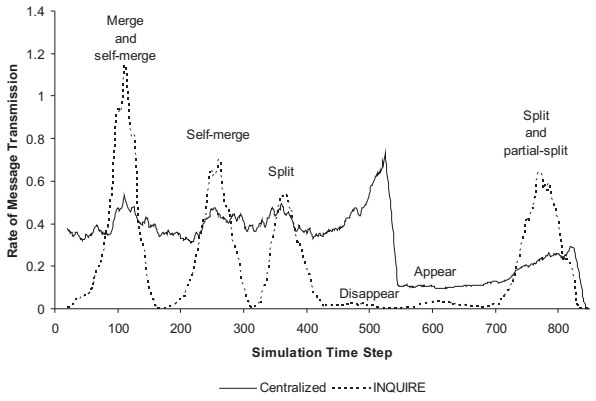

FIGURE 13. Comparison of messages transmitted for centralized and decentralized approaches

operation for a sensor node. Consequently, a key indicator of WSN performance is the level of communication overhead of an algorithm. Thus, in addition to verifying INQUIRE's correctness, the algorithm's efficiency was investigated in terms of the number of messages transmitted by WSN nodes. INQUIRE was compared to a centralized strategy in which a node that detects a change in sensor reading sends a message to a "sink" node. This sink node is located at the center of the network and is responsible for all processing. This strategy was chosen as the most efficient possible centralized strategy for comparison with INQUIRE, since 1) only updates are transmitted to the central node (no redundant or repeated information is ever communicated); and 2) the central node is ideally positioned to have on average the minimum possible distance to other nodes.

The results for a specific sequence of topological events are summarized in Figure 13, averaged over 410 simulation runs. The graph plots the rate of message transmission in the network against simulation time. As time passes, regions/holes appear, expand and contract, resulting in various topological changes. The curves in Figure 13 exhibit some fine-grained stochastic variability as a consequence of random differences in performance caused by different (randomized) node locations.

The specific sequence of events used to generate the example results in Figure 13 is illustrated in Figure 14. Note that while Figure 14 shows the key topological changes, the regions actually undergo continuous change, moving incrementally between the different topological states. The curve for the centralized algorithm exhibits some features as a direct consequence of the distance of activating nodes away from the central sink node. Increasing distance of the activating node from the central sink leads to an increased number of hops to transmit the message back to the sink node, thus increasing the number of messages transmitted. In the final stages of the region evolution, the number of messages for the centralized algorithm drops quite dramatically because all the activating nodes associated with the hole appearance and growth are located very close to the sink in the center of the region.

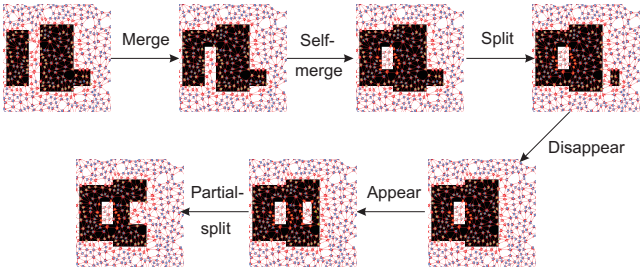

FIGURE 14. Simulation scenario

Table 3 shows the average communication costs of INQUIRE for detecting the different events, maintaining state and reporting the topological events to the sink (at an average of 5.45 messages per report). Figure 15 represents the costs graphically, showing how the rate of message transmission increases as the ratio of occurrence of a topological event to non-topological event increases.

6.1. Discussion

INQUIRE has access to the same information as a centralized solution, and so is equally accurate as a centralized solution. The results demonstrate that INQUIRE is more efficient than a centralized alternative in terms of communication scalability. The communication corresponding to merge, self-merge, split, and partial-split events is comparatively costly, leading to the spikes in Figure 13; this finding is confirmed in Figure 15. The self-merge event is the

TABLE 3. Communication costs of detecting spatial events, including reporting. In contrast, it takes 5.45 messages to report to a base station.

Event	Average number of messages
Expansion	1
Contraction	1
Appearance	5.45
Disappearance	5.45
Merge	22.1
Split	24.8
Self-merge	31.2
Partial-split	11.3

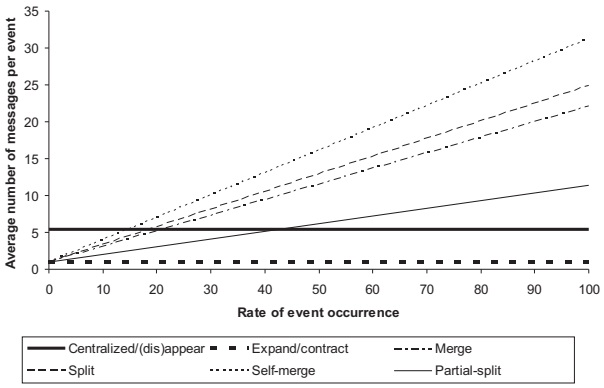

FIGURE 15. Limit of efficiency of INQUIRE. The horizontal axis shows, given 100 changes, how many of them are the named event rather than being expansion/contraction.

least efficient to manage because the procedure requires a test for containment between the two boundaries, in our implementation by computation of the area of regions enclosed by the two boundaries (see Section 5.4). INQUIRE is more efficient than a centralized approach as long as at most 14.8% of the changes are self-merge events. At the other end of the spectrum, partial-split events are the most efficient to manage (as long as at most 43.2% of the changes are partial-split events) because the updates only involve the hole that is merging with the outside. Overall, a high frequency of expansions/contractions compared to the topological events will make INQUIRE more efficient than the centralized approach.

Our experiments also provide evidence that as the density of nodes is increased, INQUIRE's performance improves compared to the centralized approach. Figure 16 shows an apparent increase in scalability, indicated by a decrease in messages sent by INQUIRE as the network size increases, when compared with a centralized approach. Although the determination coefficient of the regression is low, $R^2 = 0.29$, further investigation found this could be attributed to high variability in the number of messages generated following a split event when updating around boundaries of different sizes. In cases where regions split into two components of comparable size, the fit improved to $R^2 = 0.88$.

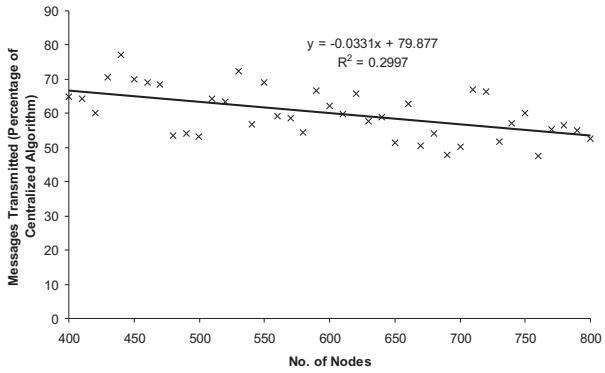

FIGURE 16. Ratio of total message transmission by INQUIRE compared to centralized approach

In summary, the results provide strong evidence that INQUIRE is indeed more scalable than a centralized solution.

6.2. Varying the order of events

In the experiment above, there is a fixed pattern of events: merge → self-merge → split → disappear → appear → partial-split. To demonstrate that the order of events does not influence the results, many other experiments with different orderings of events were performed. One example of such an experiment is in Figure 17, where the order of events is appear → merge → self-merge → partial-split → split → disappear. The results for this experiment are presented in Table 4.

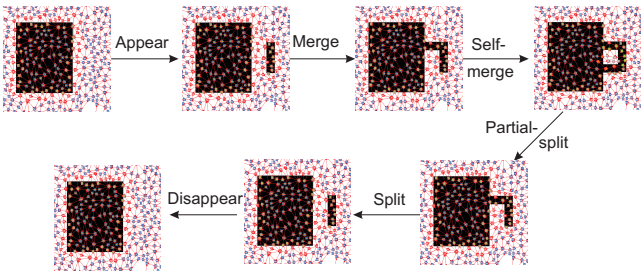

FIGURE 17. Experiment with changed order of events.

In this second experiment the region sizes and location of the events are different to that in the first experiment. These give rise to the difference in values between Tables 3 and 4 are due to these variations. However, the experiments show that the *relative* costs of monitoring the events are unchanged. Also, INQUIRE was again more efficient than the centralized approach, requiring on average 35% of the messages that the centralized approach needed.

TABLE 4. Communication costs of detecting spatial events, including reporting. In contrast, it takes 7.62 messages to report to a base station.

Event	Average number of messages
Expansion	1
Contraction	1
Appearance	7.62
Disappearance	7.62
Merge	15.2
Split	18.3
Self-merge	32.5
Partial-split	12.4

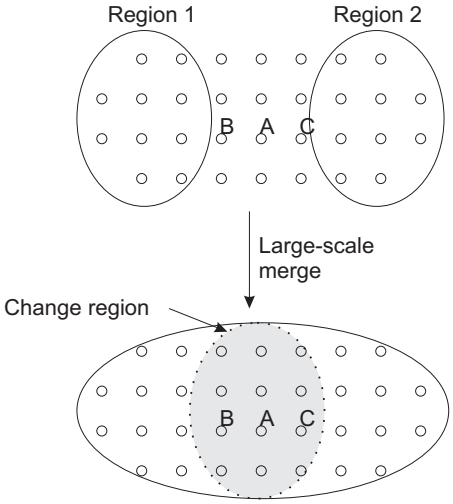

FIGURE 18. For a non-incremental change, a leader is elected and information is passed to it so that it can make a decision.

7. ISSUES

To simplify the presentation of the algorithm, the preceding section did not directly address four important issues: non-incremental changes; network granularity; initialization of network state; and generating unique region labels. This section provides a more detailed discussion of these issues.

7.1. Non-incremental changes

One of the major issues concerning the research in this paper is that it only deals with incremental changes. In environments that change slowly, or with high sensing rates, incremental change may be realistic. However, a deployable algorithm will need to cater for non-incremental change.

One way that INQUIRE can be applied to large-scale changes is by treating the area of change as a single automaton. This can be done by electing a leader in the change region that collects information from boundary nodes in the region. An example is presented in Figure 18, where Nodes B and C are adjacent to Regions 1 and 2 respectively. When a non-incremental merge occurs, the nodes in the merge region select A to be the leader and inform it of their status. In this manner, A learns that the change region is adjacent to two different regions, so, according to the algorithm in Section 5.4, it determines that a merge has occurred.

7.2. Network granularity

Most simple sensors take point readings of their environment. In such cases, it is not possible to make readings of the environment *between* sensor nodes. In order to construct a picture of the continuous areal objects they are monitoring, it is necessary to make some assumptions about these unknown readings. In INQUIRE, this is achieved qualitatively using the neighborhood structure. Where two neighboring nodes sense different readings, they infer that there must exist a boundary between them (even though they need make no attempt to interpolate *where* that boundary might lie). Conversely, where two neighboring nodes have the same reading, they may reasonably assume that there exists no boundary in between.

In any WSN deployment, there may be finer-grained detail that the nodes cannot detect. However, it is important to note that this is not a limitation of the WSN; any measurement equipment will exhibit limited granularity and so miss finer-grained change (in the same way, for example, that remote sensing cannot detect sub-pixel variation).

Similarly, in addition to spatial granularity, this work assumes relatively fine temporal granularity, such that sensor readings are sufficiently close in time to detect all salient changes. It may be possible that finer-grained changes are occurring in between coarser-grained sensor observations, but again such limitations exist for any measurement equipment, and are not unique to WSN (cf. fine-grained changes that occur between successive remotely sensed images).

7.3. Initialization of state

The algorithm presented relies on maintaining the boundary state of all nodes. In the experiments (Section 6), all nodes are initialized with non-boundary states, and an areal object is grown incrementally from a blank field.

In practice, however, it is likely that sensors will be deployed in environments with already existing areal objects. In such cases, the nodes will have to initialize their states by determining which regions/holes are containing which. Similarly, it may be desirable to periodically perform error checking during run time to validate and stabilize the algorithm.

Mechanisms for determining initial state and error-checking are not addressed in this paper, but a solution to this problem has already been devised, presented in [24]. In short, the INQUIRE algorithm defines

a *long-running* query, resident in the network; by contrast [24] defines a *snapshot* query, which generates a one-off result. The intuition behind the procedure used in [24] is to combine and adapt four existing techniques. First, a variation of face-routing [34, 35] is used to route information around region and hole components; second, leader election in a ring [41] is used to identify a leader for the ring; third, decentralized area computation [25] is used by each leader to determine a consistent orientation for regions; finally, georouting [42] is used to determine containment between region and hole components. Taken together, the approach is akin to a decentralized version of the classic semi-line point-in-polygon algorithm in computational geometry [40]. The algorithm is shown to be $O(|V|)$ communication complexity, but does require that nodes are aware of their coordinate positions (although the approach is highly robust to inaccuracy in node coordinates [24]).

7.4. Unique region labels

Region labels are required for a variety of reasons, including for locally determining whether a merge has occurred. As long as the activating node s is outer to more than one region label ($|OutN(s,t)| \geq 1$), it knows that region components/holes have merged. For this method to be effective, the region labels must be unique. To achieve unique region labels, the simulations adopt a simple heuristic that labels are a combination of the time stamp and the identifier of the node where the region component/hole appeared (or the elected node during split and self-merge). In practice, there may be many other ways to ensure unique labels, for example using some other combination of information available to a node, including its location, its neighbors, and so on.

8. CONCLUSION

This paper takes a conceptual model of areal objects and presents a method for distributing and using it in a sensor network. Each node in the network knows only information it has sensed, or that has been explicitly communicated to it. No single node is aware of the global image of the areal object or the global state of the network. Using this distribution of information, INQUIRE has been developed for the decentralized monitoring of region evolution by a WSN.

The simulation results show that this decentralized approach allows nodes to locally detect spatiotemporal changes of areal objects. INQUIRE uses only the information from direct neighbors of the activating node to identify the type of change occurring. For the events of merge, split, self-merge, and partial-split, after the local detection of the event, there is a state maintenance phase that involves nodes at the region's boundary. However, when the majority of events are expansions/contractions, which are unreported by INQUIRE, it is more efficient than a centralized approach, since the latter must communicate *all* changes. If a field involves frequent topological events, a centralized approach would be more suitable in terms of communication. However, in most applications it is to be expected that topological events will be greatly outnumbered by non-topological expansion and contraction events.

There are multiple avenues for further research into this area, such as addressing non-incremental change (see Section 7.1). Additionally, other issues that need to be addressed include node failures (which can be robustly handed by regarding them as changes in boundary conditions), message loss, and fluctuations of topology (such as one region splitting and merging again frequently).

One avenue for further investigation is the performance of INQUIRE in the face of inaccurate (location or environmental) sensors. As already mentioned, to a degree it is expected that INQUIRE is robust to such inaccuracies because it uses qualitative location and location-based descriptors (such as adjacency and relative area). Indeed, any of the algorithms that exist for identifying topological change, centralized and decentralized, will degrade in performance with increasing sensor or localization. Initial work has already begun using a three-valued representation of boundary (inside, outside, and indeterminate) in an attempt to increase further robustness to sensor errors [43].

ACKNOWLEDGMENTS

Dr Duckham is the recipient of an Australian Research Council Future Fellowship (project number FT0990531). This material is partly based upon work supported by the US National Science Foundation under Grant number IIS-0916219. We are grateful to the editor and three anonymous reviewers who provided the detailed, challenging, and constructive comments that formed the basis of our revisions.

REFERENCES

[1] Szewczyk, R., Osterweil, E., Polastre, J., Hamilton, M., Mainwaring, A., and Estrin, D. (2004) Habitat monitoring with sensor networks. *Communications of the ACM*, **47**, 34–40.

[2] Hamilton, M. P., Graham, E. A., Rundel, P. W., Allen, M. F., Kaiser, W. J., Hansen, M. H., and Estrin, D. L. (2007) New approaches in embedded networked sensing for terrestrial ecological observatories. *Environmental Engineering Science*, **24**, 192–204.

[3] Hart, J. K. and Martinez, K. (2006) Environmental sensor networks: A revolution in the earth system science? *Earth-Science Reviews*, **78**, 177–191.

[4] Lynch, N. A. (1996) *Distributed Algorithms*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[5] Arms, S., Townsend, C., Churchill, D., Galbreath, J., and Mundell, S. (2005) Power management for

energy harvesting wireless sensors. *SPIE International Symposium on Smart Structures & Smart Materials.*

[6] Beal, J. and Schantz, R. (2010) A spatial computing approach to distributed algorithms. *45th Asilomar Conference on Signals, Systems, and Computers.*

[7] Kulik, J., Rabiner, W., and Balakrishnan, H. (1999) Adaptive protocols for information dissemination in wireless sensor networks. *Proc. 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, Seattle, WA, pp. 174–185. ACM.

[8] Datta, S., Bhaduri, K., Giannella, C., Wolff, R., and Kargupta, H. (2006) Distributed data mining in peer-to-peer networks. *IEEE Internet Computing*, **10**, 18–26.

[9] Aguilar, J. and Gelenbe, E. (1997) Task assignment and transaction clustering heuristics for distributed systems. *Information Sciences*, **97**, 199–219.

[10] Gelenbe, E. and Timotheou, S. (2008) Random neural networks with synchronized interactions. *Neural Computation*, **20**, 2308–2324.

[11] Georgiopoulos, M., Li, C., and Kocak, T. (2011) Learning in the feed-forward random neural network: A critical review. *Performance Evaluation*, **68**, 361–384.

[12] Greenstein, B., Kohler, E., Culler, D., and Estrin, D. (2004) Distributed techniques for area computation in sensor networks. *LCN '04: Proc. 29th Annual IEEE International Conference on Local Computer Networks (LCN'04)*, Washington, DC, USA, pp. 533–541. IEEE Computer Society.

[13] Nowak, R. and Mitra, U. (2003) Boundary estimation in sensor networks: Theory and methods. *Proc. 2nd International Workshop on Information Processing in Sensor Networks (IPSN'03)*, pp. 80–95.

[14] Hellerstein, J. M., Hong, W., Madden, S., and Stanek, K. (2003) Beyond average: Towards sophisticated sensing with queries. *Proc. 2nd International Workshop on Information Processing in Sensor Networks (IPSN'03)*, pp. 63–79. Springer.

[15] Williams, D. and Shah, M. (1992) A fast algorithm for active contours and curvature estimation. *CVGIP: Image Understanding*, **55**, 14–26.

[16] Jin, G. and Nittel, S. (2006) NED: An efficient noise-tolerant event and event boundary detection algorithm in wireless sensor networks. *Proc. 7th International Conference on Mobile Data Management (MDM'06)*, Washington, DC, USA 153. IEEE Computer Society.

[17] Chintalapudi, K. and Govindan, R. (2003) Localized edge detection in sensor fields. *Ad Hoc Networks*, **1**, 273–291.

[18] Zhao, W. and Liang, Y. (2011) Kernel-based Markov random fields learning for wireless sensor networks. *Proc. Conference on Local Computer Networks (LCN)*, pp. 155–158.

[19] Smyrnakis, M. and Leslie, D. (2010) Dynamic opponent modeling in fictitious play. *Computer Journal*, **53**, 1344–1359.

[20] Lian, J., Chen, L., Naik, K., Liu, Y., and Agnew, G. B. (2007) Gradient boundary detection for time series snapshot construction in sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, **18**, 1462–1475.

[21] Worboys, M. F. and Duckham, M. (2006) Monitoring Qualitative Spatiotemporal Change for Geosensor Networks. *International Journal of Geographical Information Science*, **20**, 1087–1108.

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

[23] Farah, C., Schwaner, F., Abedi, A., and Worboys, M. (2011) A distributed homology algorithm to detect topological events via wireless sensor networks. *IET Journal of Wireless Sensor Systems*, **1**, 123–178.

[24] Duckham, M., Nussbaum, D., Sack, J.-R., and Santoro, N. (2011) Efficient, decentralized computation of the topology of spatial regions. *IEEE Transactions on Computers*, **60**, 1100–1113.

[25] Sadeq, M. and Duckham, M. (2009) Decentralized area computation for spatial regions. *Proc. SIGSPATIAL ACMGIS*, New York, pp. 432–435. ACM.

[26] Sadeq, M. J. and Duckham, M. (2008) Effect of neighborhood on in-network processing in sensor networks. In Cova, T., Beard, K., Goodchild, M., and Frank, A. U. (eds.), *Geographic Information Science*, Lecture Notes in Computer Science, **5266**, pp. 133–150. Springer, Berlin. (Conference accepted 31% of submitted papers).

[27] Buneman, O. (1970) A grammar for the topological analysis of plane figures. In Meltzer, B. and Michie, D. (eds.), *Machine Intelligence*, pp. 383—393. Elsevier.

[28] Worboys, M. and Bofakos, P. (1993) A canonical model for a class of areal spatial objects. *Proc. Third International Symposium on Advances in Spatial Databases (SSD'93)*, Berlin, pp. 36–52. Springer.

[29] Clementini, E., Felice, P., and Califano, G. (1995) Composite regions in topological queries. *Information Systems*, **20**, 579–594.

[30] Li, S. (2006) A complete classification of topological relations using the 9-intersection method. *International Journal of Geographical Information Science*, **20**, 589–610.

[31] Jiang, J. and Worboys, M. F. (2009) Event-based topology for dynamic planar areal object. *International Journal of Geographical Information Science*, **23**, 33–60.

[32] Kuhn, F., Wattenhofer, R., Zhang, Y., and Zollinger, A. (2003) Geometric ad-hoc routing: Of theory and practice. *Proc. 22nd ACM International Symposium on the Principles of Distributed Computing (PODC)*, pp. 63–72.

[33] Zhao, F. and Guibas, L. (2004) *Wireless Sensor Networks: An Information Processing Approach.* Elsevier/Morgan-Kaufmann, Amsterdam.

[34] Karp, B. and Kung, H. (2000) GPSR: Greedy perimeter stateless routing for wireless networks. *Proc. 6th annual international conference on Mobile computing and networking (ACM/IEEE MobiCom)*, Boston, MA. ACM.

[35] Bose, P., Morin, P., Stojmenovic, I., and Urrutia, J. (1999) Routing with guaranteed delivery in ad hoc wireless networks. *Proc. 3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pp. 48–55.

[36] De Silva, V. and Ghrist, R. (2006) Coordinate-free coverage in sensor networks with controlled boundaries via homology. *International Journal of Robotics Research*, **25**, 1205–1222.

[37] Kröller, A., Fekete, S., Pfisterer, D., and Fischer, S. (2006) Deterministic boundary recognition and topology extraction for large sensor networks. *Proc. Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1000–1009.

[38] Funke, S. (2005) Topological hole detection in wireless sensor networks and its applications. *Proc. 3rd ACM/SIGMOBILE International Workshop on Foundations of Mobile Computing (DIAL-M-POMC)*, New York, NY, pp. 44–53. ACM.

[39] Tahbaz-Salehi, A. and Jadbabaie, A. (2010) Distributed coverage verification in sensor networks without location information. *IEEE Transactions on Automatic Control*, **55**, 1837–1849.

[40] Preparata, F. P. and Shamos, M. I. (1985) *Computational Geometry: An Introduction*. Springer-Verlag New York, Inc., New York, NY, USA.

[41] Santoro, N. (2007) *Design and Analysis of Distributed Algorithms*. Wiley, New Jersey.

[42] Stojmenovic, I. (2002) Position based routing in ad hoc networks. *IEEE Communications Magazine*, **40**, 128—134.

[43] Duckham, M., Stell, J., Vasardani, M., and Worboys, M. (2010) Qualitative change to three-valued regions. In Fabrikant, S., Reichenbacher, T., van Kreveld, M., and Schlieder, C. (eds.), *GIScience*, Berlin, Lecture Notes in Computer Science, **6296**, pp. 249–263. Springer.