# Detecting Topological Change Using a Wireless Sensor Network

Christopher Farah , Cheng Zhong, Michael Worboys, Silvia Nittel

Department of Spatial Information Science and Engineering,
University of Maine, Orono, ME, 04469, USA
{cfarah, czhong, worboys, nittel} @spatial.maine.edu

**Abstract.** Dynamic geographic phenomena, such as forest fires and oil spills, can have dire environmental, sociopolitical, and economic consequences. Mitigating, if not preventing such events requires the use of advanced spatio-temporal information systems. One such system that has gained widespread interest is the *wireless sensor network* (WSN), a deployment of *sensor nodes* – tiny untethered computing devices, which run on batteries and are equipped with one or more commercial off-the-shelf or custom-made sensors and a radio transceiver. This research deals with initial attempts to detect topological changes to geographic phenomena by an environmentally deployed wireless sensor network (WSN). After providing the mathematical and technical preliminaries, we define topological change and present in-network algorithms to detect such changes and also, to manage the WSN's resources efficiently. The algorithms are compared against a resource-heavy continuous monitoring approach via simulation. The results show that two topological changes, hole loss and hole formation, can be correctly detected in-network and that energy is greatly saved by our event-driven approach. In future work, we hope to test the algorithms over a broader range of topological changes and to relax some of the network assumptions.

**Keywords.** Wireless sensor networks, distributed, algorithm, topological change, areal object.

## 1 Introduction

Dynamic geographic phenomena, such as forest fires and oil spills, can have dire environmental, sociopolitical, and economic consequences. Mitigating, if not preventing such events requires the use of advanced spatio-temporal information systems. One such system that has gained widespread interest is the *wireless sensor network* (WSN), a deployment of *sensor nodes* – tiny untethered computing devices, which run on batteries and are equipped with one or more commercial off-the-shelf or custom-made sensors and a radio transceiver. Beyond catastrophe management, it is expected that WSNs will play a key role in ubiquitous spatial computing, offering researchers and application domain specialists unprecedented opportunities in environmental sensing, monitoring, and analysis. In contrast to other sensing technologies, e.g. LIDAR, a WSN is not restricted to particular types of phenomena: a

WSN can detect any *measurand* – a physical parameter, such as light intensity, temperature, or ozone – so long as the corresponding sensor has been developed, and the device can withstand the deployment region's environmental conditions. Given a fixed deployment of sensor nodes over a geographic region, one of two monitoring paradigms can be adopted: *continuous* monitoring or *event-driven* monitoring. In the former, every sensor node in the network samples the environment at a constant rate. In the latter, a sensor node's level of activity is affected by local changes in the measurand. While continuous monitoring offers the best responsiveness possible, it does so at the cost of network resources, i.e. power consumption, and temporal resolution of the environmental data. In the case of event-driven monitoring, regions of low sensor activity demand less of the network's processing, allowing resources to be dedicated to those regions of high activity. By shutting down nodes, however, resolution and responsiveness can decrease. Thus, either approach leads to a compromise in monitoring.

In this paper, dynamic *topological events* with regard to continuous spatial phenomena, events such as hole formation and merging, govern the activity of the network. The motivation is three-fold: (1) to provide a framework for the classification of environmental phenomena by topological behavior, (2) to reduce the cost of data transmission and in-network information processing, and (3) to optimize the number of active nodes through a tiered network. To achieve these goals, nodes are assigned to *clusters* - geographic subsets that partition the region of deployment. For each cluster, one node is promoted to the rank of *cluster head*. By virtue of this hierarchy, updates can be coordinated via cluster heads, allowing the network to "keep up" with global topological changes.

In summary, the contributions of this paper include the following:

- Algorithms for topological event detection in WSNs.
- Algorithms for dynamic resource management.
- Evaluation of proposed algorithms in comparison with the continuous monitoring approach. Events are detected correctly and network energy is saved.

The remainder of the paper is organized as follows. Section 2 provides mathematical and technical preliminaries and summarizes related work. Section 3 describes our approach for event detection in dynamic sensor networks. In section 4, the algorithms for network management and event detection are described in detail. Section 5 outlines the simulation approach to test our algorithms and discusses the corresponding results. Finally, we conclude this paper and discuss future work.

## 2 Mathematical Preliminaries and Related Work

### 2.1 Mathematical Preliminaries

A *scalar field* is a spatial domain R, such that for each point $p \in R$, there is a unique scalar value, $s_p$, assigned to p. In this work, it is assumed that R is planar and that a sensor node associated with point p detects $s_p$ with complete precision. For example,

in the case of forest fire monitoring, each point in the region of network deployment resides in a scalar field where temperature is the scalar value. In order to derive topological events from the scalar field, a threshold value is designated relative to the sensor readings, invoking a Boolean response for each point in the spatial domain. Thus, the scalar field is discretized; rather than a continuous range of scalar values, there are precisely two values, 0 and 1. This discretized field is approximated by the network; the resolution depending upon the density of nodes in the WSN, and the proportion that are active. Figure 1 shows the progression from a scalar field to its discretized approximation. This particular example is derived from aerial images of an oil spill off the coast of Spain.

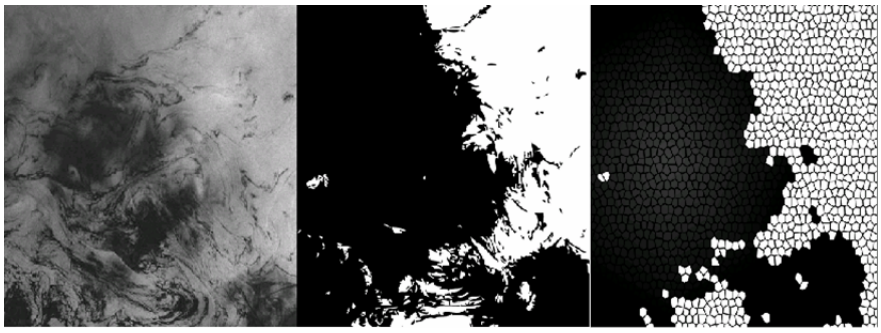

**Fig. 1.** From scalar field to the discretized, WSN approximation

Let us consider that part of the domain whose corresponding scalar values are 1. This is an areal object [1] and consists of one or more *connected components* (components for simplicity) of R, i.e. regions in which any two points in one such region can be joined by a path, completely contained in the region. Each component has topological properties that can be determined. For the purpose of this investigation, the properties of greatest interest are connectedness and *genus*, which is a count of the number of holes in a component. The motivation is simple: by keeping track of these two properties over consecutive sensor samples, six atomic topological changes can be identified. A *topological change* is a change to an areal object such that there is no homeomorphism between the areal object in its initial state and its final state. Thus, a topological change occurs if an areal object, by virtue of the discretized scalar field's evolution over time, changes status with regard to one or more topological properties. In this paper, "topological change" and "event" are synonymous. The atomic changes to be detected are:

- hole formation / hole loss
- self-split / self-merge
- split / merge

As can be seen in Figure 2, the genus has changed for the cases: hole formation, hole loss, self-split, and self-merge. For the final two topological changes, split and merge, the property connected has changed but not the genus. Jiang and Worboys [1] have proved that any topological change resulting from changes only in genus and

connectedness can be expressed as a composition of those above. Therefore, it will suffice to deal with atomic topological changes.

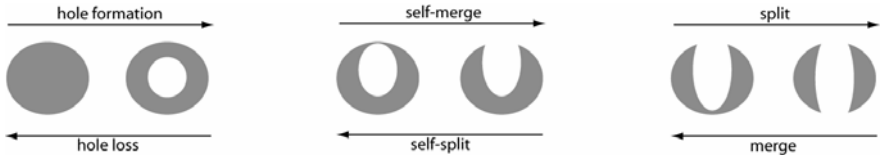


**Fig. 2.** The six primary topological changes

An *incremental change* to a WSN is the change of sensor status, relative to threshold, of a single node over the entire network at a time t. Such a change will result in zero or more topological changes. In order to capture the changes outlined above, *the neighborhood ring* is introduced. The neighborhood ring is a cyclic data structure stored at a node that maintains its nearest neighbor readings in counterclockwise order. In order to identify its neighbors, a node broadcasts a message at some user-defined fraction of the communication range of the node. Neighbors within this range will have a corresponding entry in their neighborhood ring. Let $u$ be a node and $v_1$, $v_2$, …, $v_k$ be the $k$ one-hop neighbors of $u$ within a predefined distance, sequenced in counterclockwise cyclic order around $u$, where a starting node $v_1$ is randomly assigned in advance. The neighborhood ring associated with $u$ is a ring data structure [$s_1$, $s_2$, … , $s_k$], starting from $v_1$ where $s_i$ is the sensor reading of $v_i$, which are mapped to the Boolean values 0 and 1 as previously described. Figure 3 shows a node $u$ and its neighbors, the underlying discretized scalar field in gray, and two equivalent neighborhood rings. A neighborhood ring - not unlike the connected components of the spatial domain – can be thought of as a collection of contiguous subsequences, i.e., sequences of the form [0,…1,1,…,1,…,0] or [1,…1,0,0,…0,1,…,1], since the neighborhood ring is cyclic, i.e. it can start at any neighboring node. Such a contiguous subsequence is called a *neighborhood component*, and indicates a region of similar sensing. Two neighborhood components are indicated in Figure 3 by dashed rectangles and each can be identified as a sequence of 1s in each neighborhood ring. In order to determine which topological change has occurred, a node that has changed status will initiate a series of tests based upon the neighborhood components of its neighborhood ring.

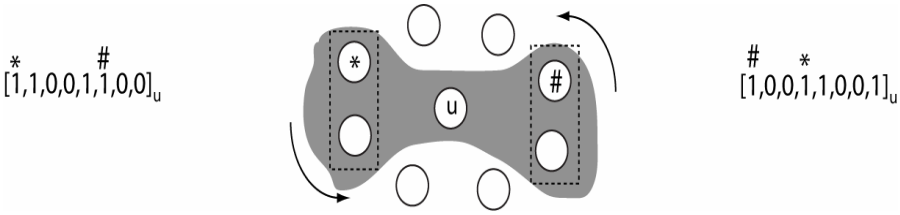


**Fig. 3.** Two equivalent neighbor rings of node u

### 2.2 Network Assumptions

A *distributed* computing environment is one in which multiple computing devices or nodes (often) operate in parallel and achieve a common goal by processing available data (when appropriate) in a cooperative fashion, thereby passing the intermediate result as a message to other nodes. Two interesting properties of a distributed WSN are: (i) there is no global clock, and (ii) only local data is stored at a node. The consequences of (i) are that: classic synchronization can not be used to order computations and many events will be temporally incomparable, i.e. it will not be possible for any node to correctly determine the order of topological events, even if the events are temporally ordered in the environment. The consequence of (ii) is that no node has a global view of the network. This does not mean global properties cannot be computed, but that no node has all of the raw data necessary to execute such a computation. While explicit reference to these properties will not be made, it is interesting to bear them in mind in the development that follows.

In addition to the generic distributed system assumptions, we make five additional assumptions related to the operation of the WSN in this research. They are:

1) Neighboring nodes of the same status belong to the same component.
2) Non-neighboring nodes of the same status belong to the same component only if there is a node path between them, such that each node on the path has the same status.
3) Each node knows its own location through either a GPS device or some GPS-less techniques [2, 3], as well as the angle-of-arrival of packets.
4) A node stores its previous and current sensor reading relative to threshold, its neighbor ring, as well as the ID.
5) A node that has been promoted to cluster head stores all data noted above, as well as the genus, the node ID of each adjacent cluster head, and any temporary data structures (See Section 3.1.) needed to complete computations.

The first two assumptions allow sensor data to infer properties of the underlying scalar field. It is easily proved that assumption (2) follows from assumption (1) but it is stated on its own for clarity. Assumption (3) ensures that each node can identify the cluster it belongs to and that a well-ordered neighbor ring can be constructed. Assumptions (4) and (5) ensure that correct topological event detection can be carried out. In particular, if a split or merge occurs, region IDs need to be updated at each node, while a genus update is required for the remaining four topological changes.

### 2.3 Related Work

#### 2.3.1 Topology Discovery

Current research in hole detection using WSNs has primarily focused on the characterization of the communication graph within the network, i.e. a graph whose vertices are sensor nodes and edges are communication links between sensor nodes. In particular, research has been conducted in order to ensure the effective routing of messages, even in the presence of holes in the communication graph. In [4], the

author presents a solution to the problem which requires more computational power than is typically assumed of sensor nodes, and therefore could not be implemented in a distributed setting. Clearly, for long-term unsupervised deployments, a more robust, decentralized solution is needed. In [5], hole detection is distributed and decentralized. However, a 'flat' routing approach is applied in order to define and update each hole's boundary. In other words, data is simply passed via nearest neighbors. While this may be feasible in maintaining the topology of a slowly changing communication graph, the topology of the underlying scalar field is likely to evolve much more rapidly. Additionally, there are different algorithmic expectations: holes in the communication graph result in node failure and therefore, a loss of spatial resolution, while holes in spatial phenomena only result in a change in sensor value, but not the failure of the sensor node itself. Therefore, more efficient and appropriate update techniques are proposed in this work.

The prerequisite for topological change detection is boundary detection in the network. Chintalapudi and Govindan [6] discuss three boundary detection methods and return sufficient data so that the base station can construct an accurate boundary. They do not however, transmit boundary information back into the network, preventing the possibility of localized, in-network updates of the boundary shape and its location. Ding et al. [7] propose localized fault-tolerant boundary and faulty sensor detection using spatial data mining techniques. These techniques have to report all boundary node information to the base station: the topological changes can only be deduced at the base station, after it generates different field snapshots using received data. As in [1], such techniques are not tractable in the case of remote deployments. In this paper, region boundary detection is based on neighboring node value differences: a boundary exists between two nodes if their sensors detect different measurand concentrations, relative to the designated threshold. Based on the detected boundary, all changes are deduced in the network and then are reported back to the base station, on user demand.

### 2.3.2 Resource Management

In WSNs, particularly for long-term deployments, energy conservation is critical. If communication is not dealt with carefully, then network resources can be unnecessarily expended, which is the largest energy consumer in a sensor network. For example, even if a node broadcasts data intended for a few selected neighbors, every node within transmission range and operating on the same channel must receive and process each packet, whether the packet has computational importance or not. Chen et al. [8] show that the energy consumption ratio, idle:receive:transmit is 1: 2: 2.5. This observation motivates approaches that either reduce the number of active nodes or reduce node contention. Xu et al. [9] develop a geographic adaptive fidelity (GAF) algorithm, which can be implemented in conjunction with any ad-hoc routing algorithm. GAF identifies "equivalent nodes" from a routing perspective: two nodes are equivalent if the cost of routing messages through one is the same as the other. Thus, the network is partitioned into virtual grids. Within each grid, most nodes can be set to sleep, so long as they are not a source, sink or critical intermediate node within the routing chain. Simulation results demonstrate savings of 40-60% as compared with unmodified ad-hoc routing protocol. Zhang and Cao propose DCTC [10] for target tracking in WSNs. A tree structure is constructed for moving target

tracking. It uses a prediction-based schema for tree expansion and pruning. Only nodes near the target are activated. While effective for target tracking, this approach does not address topological changes to the discrete scalar field. In [11], an advanced sweep algorithm is implemented in order to activate sensor nodes in front of an event wavefront and to deactivate sensor nodes behind the wavefront. The algorithm is implemented in a small network consisting of Mica2 MOTES and demonstrates effective detection of the wavefront while conserving network resources. However, as the author states, the topological sweep algorithm does not admit a distributed approach. Duckham et al. [12] describe a triangulation approach for monitoring dynamic fields. The sensor network is triangulated and most sensors that are not in the event regions are deactivated.

In this paper, a combination of traditional clustering is used in conjunction with GAF in order to ensure network energy conservation without compromising event detection. Specifically, sensors along the boundary of different regions are activated to detect the boundary's evolution, while sensors that are in low activity regions and are not critical in terms of routing are deactivated. When coupled with our local topology discovery approach, the efficient reporting of topological change becomes feasible.

# 3 Event Detection in Sensor Networks

## 3.1 Event Detection

The goal of this research is to efficiently detect the topological changes outlined in 2.2, through distributed computations in a distributed WSN. Each of the topological changes implies that the corresponding phenomena's boundary has undergone a dynamic change. The node set approximating the boundary will also change, so long as network resolution is sufficient. It follows that those incremental changes of importance occur at the boundary or create a new boundary. Thus, changes to boundaries allow each of the topological changes to be detected. For the sake of simplicity, only incremental changes will be tested via simulation. This can be justified since: (1) the change in sensor value (relative to threshold) of a single node can result in any of the six topological changes under investigation, and (2) methods to compute non-incremental change as an extension of incremental change will be outlined.

In Figure 4, two regions, $R_1$ and $R_2$, are illustrated, along with three nodes that have changed status and must compute if a topological change has occurred. Assuming that each node stores the ID of the region it belongs to, in the best case, the outcome can be determined on the basis of the neighborhood ring and subsequent neighbor components. This is true for hole formation, hole loss, merging, and self-merging. Consider node *u*, shown in Figure 4. As a hole forms in the region monitored by the node, its sensor detects this change, and hence node *u* uses its neighborhood ring to determine the kind of change. Since its neighborhood ring is of the form [1,1,…,1], even without region ID, node *u* identifies that the change is a hole

formation. Hole loss is similar, except that node *u* proceeds from a sensor value of 1 to 0. Node *v* illustrates a merge, which can be determined by the combined facts that: (1) the neighborhood ring consists of multiple neighborhood components (above threshold) and (2) the ID differs between two or more components. If region ID were the same, then the change would be a self-merge. In the case of splitting and self-splitting, region ID and neighbor data is not sufficient. Consider node *w* as an example. By observation, a self-split is taking place since node *w*'s sensor status change has not split the region into multiple components. Node *w* however, will not know this until it confers with the nodes of this region. In contrast to a merge or self-merge, prior to a split or self-split, all nodes belonging to the region will store the same region ID ($R_1$ in this case) regardless of the event type. So, node *w* must pass a message through the region to determine if it has an *irreducible cycle* – a broadcast cycle passing through nodes of status 1, such that the cycle can't be trivially reduced without passing through a node of status 0. In particular, a cycle discovery message is passed to one node of each neighborhood component. Each receiving node in turn passes the message to a node in each of its neighborhood components. If a node receives messages originating from different neighborhood components of the initiating node, then an irreducible cycle exists. Part of this process is illustrated by the smaller, transparent circles in the figure. In this case, Node x receives messages from different neighborhood components of Node w. Thus, an irreducible cycle has been identified and therefore, the event is a self-split. Node x of course must relay this to Node w.

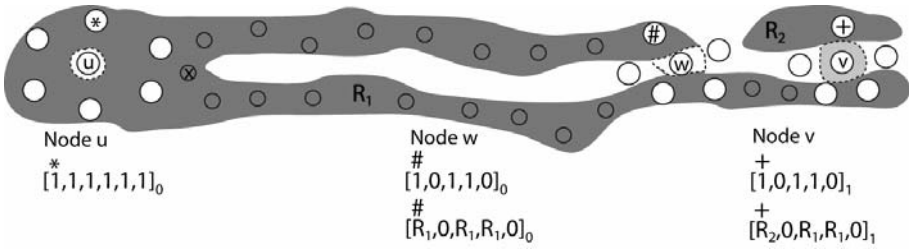


**Fig. 4.** One topological change from each of the three categories

This process is repeated until a node can no longer propagate the message, or a node has received multiple messages. If the latter occurs, as it would above, then an irreducible cycle exists. Thus, the topological change is a self-split. Otherwise, the change is a split.

In the case of non-incremental changes, the network is prone to non-scalable behavior and data conflict. To emphasize this, let us compare a non-incremental change without the use of assumption (5) to the same change with the use of assumption (5). We will refer to Figure 5, which displays nodes u and v that have changed status; cluster heads A, B, C ,and D; unnamed nodes; and communication links, labeled with the transmission round. A transmission round such as $2_v$ indicates the second round of transmission relative to node v. If there is no subscript, it indicates that the order of transmission has been coordinated by the collaborating cluster heads. A network without cluster heads is illustrated in the left pane of Figure

5. Here, nodes *u* and *v* change status, compute any topological changes, and pass the update through the component. The first two rounds of message passing are shown. Two problems arise. The first is that the number of nodes requiring updated genus data is significant: all of the nodes in the grey region. The second problem is that intermediate nodes (lying between the rippled lines) will receive conflicting data regarding the topological state of the region: node *v* reports the loss of a hole while node *u* reports the gain of a hole. By admitting assumption (5), the network is clustered as in the right pane of Figure (5). Node *u* reports a change to its cluster head *A*, which in turn reports to the affected adjacent cluster heads *B*, *C*, and *D*. Node *v* senses a change and reports this to cluster head *C*. Cluster head *C* reports one required update to node *A*, while cluster heads *B* and *D* broadcast no such report. Cluster head *A* orders the queue, first by cluster {A, C}, then by node – in this case {*u*, *v*} – and passes the data to node *u* in the third round. Node *u* makes a partial computation, passes it to cluster head *A*, which in turn passes the partial result to cluster head *C*. Cluster head *C* passes the computation to node *v*, which completes the computation, and unicasts the final result to cluster head *C*. By the eighth round, cluster head *C* multicasts the updated genus to cluster heads *A*, *B*, and *D*. The first eight rounds of message passing are shown in the figure. In order to limit network noise, the multi-channel capability of the sensor nodes is exploited: in addition to the general broadcast channel, there is a cluster-head channel, a channel for each sensor value, and four channels assigned to clusters, so that no two adjacent clusters have to operate on the same bandwidth. As a result, unsolicited nodes do not receive broadcasts, and therefore, do not waste energy on processing packets.

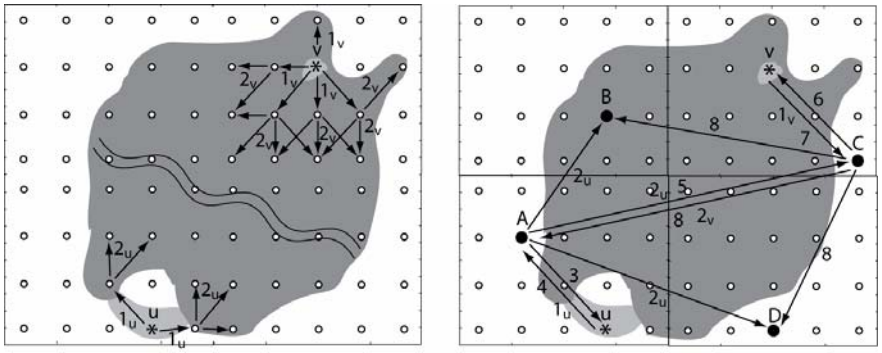


**Fig. 5.** A flat approach on the left and a tiered approach on the right

Since each cluster in the tiered approach behaves like a node in the flat approach, a 2-level network is “more scalable” but not truly scalable: cluster size, and the number of tiers are governed by the expected “size” of the topological events. This will be investigated in greater detail in future work in order to correlate event size with required level of network tiering.

### 3.2 Resource Management

Since the WSN's nodes are battery powered and have limited processing capability, economizing network resources is key. However, such economization should not undermine the network's key goal: topological event detection. To ensure a balance between event detection and resource economization, the network must be event-driven, thereby allowing for increased data resolution in areas of topological activity. While a continuous-monitoring network cannot vary its resolution or consumption of network resources, it is capable of responding to changes rapidly, since all nodes are active. The challenge in an event-driven network is to find an acceptable level of responsiveness.

In order to meet the needs outlined above, the network is tiered. We design a 2-level hierarchy network, as in the right pane of Figure 5. The upper tier network consists of cluster heads. The lower tier network is composed of all nodes in each cluster. If there are no events near a cluster, only the cluster head is active. All other nodes in the cluster can be in sleep mode, thereby conserving energy. Each pair of nodes in adjacent clusters is reachable to each other, ensured by transmission range setting.

We separate the whole network into uniform, rectangular clusters. The diagonal of each cluster is less than half the radio range in order to ensure each pair of nodes in adjacent clusters has the ability to communicate with each other. The clusters are fixed after network initialization. Assuming nodes are GPS-equipped, they can compute their geographic position, and therefore, the cluster they lie in. Each cluster has a designated cluster head, which is responsible for node activation. A node can deactivate itself if it does not detect any interesting activity.

Each node in the network can be in three states. In the sleeping state, a node does not send or receive any messages from others. This brings the most power saving to the network. In the listening state, a node can receive its neighbors' messages to determine if it should turn itself to active state but does not transmit messages. In the active state, a node collaborates with other active neighbors to monitor physical events and changes. The state transition graph of a node is shown in Figure 6. In Figure 6: (0) A node boots up. (1) A node periodically changes from active to listening mode, if there is no reading difference between the node and its neighbors over a time $t_1$. (2) A node periodically changes from listening to sleeping mode after a predefined time $t_2$. (3) A node changes to active state after it receives an activate message. (4) A node goes to listening after sleeping for a time $t_3$. More details about the node activation and deactivation will be explained in section 4.2.

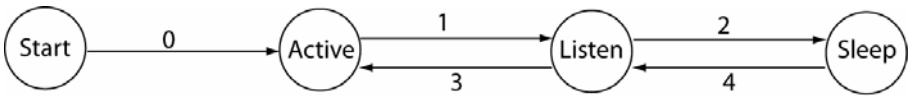


**Fig. 6.** State change graph

# 4 Algorithms

Three algorithms will be presented, covering: cluster head election, node management, and topological event detection.

## 4.1 Cluster Head Election

We use a randomized algorithm to select cluster heads. Each node in a cluster randomizes a short timer [13]. After the time out, if the node does not receive a suppression message from a cluster head, it becomes the cluster head and broadcasts a suppression message to inform all other nodes in the cluster. Since the cluster head is the only activated node in a cluster, its battery will discharge more quickly than other nodes' batteries. Thus, we promote a new cluster head after the battery power is lower than a designated threshold. If a cluster head's power drops below a threshold value, it periodically sends a request message with its residual energy information to all neighbors in the cluster. All nodes in listening state can receive this message. After receiving the message, a node that has more energy will randomize a short timer. After the time out, if it does not receive a new suppression message from other nodes, it becomes the new cluster head and broadcast a suppression message to all other nodes in the cluster.

## 4.2 Node Activation/Deactivation

Each node in a cluster is activated by the cluster head and it is self-deactivated based on the reading difference. At initialization, all nodes are in active state. Each cluster head also receives readings from, and broadcasts readings to, neighboring cluster heads. If they are all above - or below - the threshold value, no change is detected and the corresponding cluster nodes will not be activated. If a neighboring cluster head has a different Boolean response relative to threshold, then an event boundary must exist between them. In this case, the cluster head broadcasts messages to activate all nodes in its cluster. Each node, after entering the listening mode, can receive the activation request and turn itself to active state. A node in the active state will periodically compare its reading to neighbors. If there are some differences between them, it remains active for event detection. Otherwise, if there is no difference (no change) after a user specified period, it enters to listening mode for a while. If it does not receive further activation messages, it proceeds to sleep state, and then periodically alternates between listen and sleep states. The algorithm for node activation and deactivation is shown in algorithm 1.

```
While (1)
{
  if (in sleeping mode)
     After a short time T3, go to listening state;
  else if (in listening mode)
     if (receives an active message)
        Go to active state;
```

```
    else
       Go to sleeping state after a short time T2;
  else if (in active mode)
    if (no reading differences between any neighbors)
       Go to listening mode after a short time T1;
}
```

**Algorithm 1**. Non-cluster node activation and deactivation

### 4.3 Event Detection

In this section, we discuss the algorithm for event detection. If a node changes status with respect to threshold, it tests for the easiest topological changes first, and then proceeds in order of difficulty. If the neighborhood ring is uniform, then the topological change is either hole loss or hole formation. If however, the neighborhood ring is not uniform, the node checks first for non-topological changes (not outlined in this paper), and then for the topological changes merge and self-merge. If neither event is possible, the node broadcasts a message in order to discover any irreducible cycles. If there are such cycles, the event is a self-split, and otherwise, the event is a split. It should be noted that the algorithm is written for nodes that do not lie on the boundary of the network. While the modifications necessary to include boundary nodes are reasonable, they are not included below.

```
While (1)
{
  if (sensor status changes)
    Check neighborhood ring;
    if (uniform)
      if (sensor status of node is 1)
         Event: hole loss
      else
         Event: hole formation
    elseif (one neighborhood component)
         Non-topological event
    else
       Check neighborhood components' region IDs
       if (multiple region IDs)
          Event: merge
       else if (one region ID)
          if (sensor status of node is 1)
             Event: self-merge
          else
              Check for irreducible cycles
              if (irreducible cycles)
                 Event: self-split
              else
                 Event: split
}
```

**Algorithm 2.** Event detection algorithm

# 5 Simulation and Discussion of Results

## 5.1 Simulation

We evaluated our algorithms using NS2 [14], open source, network simulation software. NS2 contains standard API that facilitates the development of a network model at the network level, the node level, and the process level. In addition, many research groups have made their custom node and network models available, thereby expediting the development of future work, such as ours. In the simulation, a 100m by 80m WSN that consists of 400 nodes with 20 clusters is constructed. Each cluster is a 20m by 20m grid and there are 20 evenly distributed nodes in each. As specified in section 3, we assume nodes in adjacent clusters can communicate directly. In order to achieve this assumption, the node communication radius is set to 57m. Each node saves its nearest 8 neighbors' information around it in the neighborhood ring. The event-detection algorithm is applied in conjunction with both continuous and event-driven monitoring approaches. In the case of continuous monitoring, each node is active throughout the entire simulation interval. As discussed in the related work, even an idle node consumes a good deal of energy. Hence, the greatest savings only result from deactivating nodes. Thus, to measure network savings, it suffices to determine, and compare, the percentage of active nodes between the two approaches. Furthermore, the time taken to detect topological change is computed as a metric for network responsiveness.

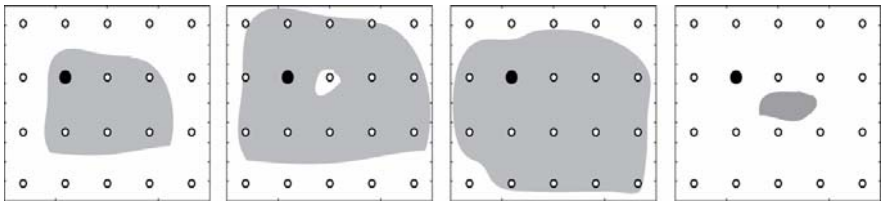

**Fig. 7.** One cluster at t=10s, 20s, 30s, and 45s

**Table 1.** Active node count at t=10s, 20s, 30s, and 45s

| | **Continuous approach** | **Event-driven approach** |
|---|---|---|
| T=10s | 400 | 191 |
| T=20s | 400 | 191 |
| T=30s | 400 | 191 |
| T=45s | 400 | 20 |

Over the course of a single simulation run, the network is initialized at t=1s. An areal object forms in the middle of the field at t=2s and continues to grow. A hole emerges in the areal object at t = 20s, and grows until t=25s, when it begins to shrink. At t=30s, the hole disappears. Then, the areal object continues to shrink and disappears at t=44s. In this scenario, sensor nodes that are active sample the environment once per second. From the simulation results, the areal object which forms at t=2s is detected at t=4s, after it covers a cluster head. After detecting the

areal object, the cluster head activates all nodes in its cluster and notifies adjacent cluster heads. Other topological changes, the hole formation at t=20s, the hole loss at t=30s, and the disappearance of the areal object at t=44s are all detected immediately. In the continuous approach, all topological changes are detected immediately after they happen. We also compare the number of active nodes between the continuous and event-driven approaches at some time snapshots t=10s, 20s, 30s, and 45s. The results are shown in Table 1. Clearly, the continuous monitoring approach keeps all nodes in active state, but the event-driven approach keeps only a few nodes in active state all the time, which means larger energy savings for the entire network.

### 5.2 Discussion

In the simulation, all topological changes are detected correctly in the network, which means our event detection algorithms works for both event-driven approach and continuous approach. Compared to previous base station based approaches [4, 11], such in-network detection is one of the most significant contributions of our work. Node will report changes to the base station which need no future process at the base station side.

By the event-driven approach, as indicated by the simulation results, the areal object appears at t=2s, but it was not detected before t=4s when the areal object covers a cluster head, which means the event-driven approach may decrease the responsiveness of the network. In comparison, the continuous approach allows for instant detection at the cost of network resources. While an event-driven approach is necessary to handle lengthy, remote deployments, it is inevitable that energy conservation sacrifices resolution and responsiveness to some degree. In particular, when a topological change occurs in a cluster that has been set to sleep, it will go unnoticed until the cluster head itself detects it. Clearly, as the cluster size decreases, responsiveness and resolution increase. However, network resources would be compromised, as in the continuous monitoring case. There should be a trade-off between responsiveness and the energy consumption. If we need better responsiveness, we can set small cluster size. If the responsiveness is not very crucial, a larger cluster size could be used. Bounds on cluster size relative to event detection will be addressed more thoroughly in future work.

## 6 Conclusion and Future Work

This paper presents algorithms for topological change detection in a WSN using event driven approaches. Different from previous approaches, we focus on detecting topological changes of areal objects monitored by the WSN. A neighborhood ring data structure is proposed for in-network event detection. By our event detection algorithms, topological changes can be detected directly in the network, other than computed at the base station after receiving all reporting messages. Each node does not have to send readings to the base station for processing. This characteristic is one of the most significant differences compared to traditional approaches. By our event-driven approach for network management, not all nodes are required to be in the

active mode all the time and then the network energy is saved greatly. The simulation results show that our event-driven approach deactivates some nodes in the network without decrease responsiveness significantly. The event detection algorithms proposed also detect all topological changes correctly in the network.

Our current simulation does not include split and self-split and the detection of such changes need global broadcasting using current algorithms via the irreducible cycles. In future work, the topological changes split and self-split will be addressed and we are trying to reduce such global broadcasting. Furthermore, non-incremental changes need to be simulated to confirm that a tiered approach effectively handles more complicated, non-atomic topological changes. This would include the correct assignment of a component's ID, as well as passing off partial computations between clusters.

**Acknowledgements.** This material is based upon the work supported by the US National Science Foundation under Grant Nos. IIS-0534429 and IIS-0429644.

## References

1. Jiang, J., Worboys, M.: Event-based topology for dynamic planar areal Object. Technical report, University of Maine (2007)
2. Cheng, X., Thaeler, A., Xue, G., Chen, D.: TPS: A time-based positioning scheme for outdoor wireless sensor networks. In: INFOCOM, pp. 268--2696 (2004)
3. Shang, Y., Ruml, W., Zhang, Y., Fromherz, M.P.J.: Localization from mere connectivity. In: MobiHoc, pp. 201--212(2003)
4. Funke, S.: Topological Hole Detection in Wireless Sensor Networks and its Applications. In: Proceedings of 3rd ACM/SIGMOBILE International Workshop on Foundations of Mobile Computing (DIALM-POMC), Cologne (2005)
5. Fang, Q., Gao, J., Guibas, L.: Locating and Bypassing Holes in Sensor Networks. Mobile Networks and Applications, vol. 11, pp. 187--200 (2006)
6. Chintalapudi, K., Govindan, R.: Localized edge detection in sensor fields. Ad Hoc Networks, vol. 1, no. 2-3, pp. 273--291 (2003)
7. Ding, M., Chen, D., Xing, K., Cheng, X.: Localized fault-tolerant event boundary detection in sensor networks. INFOCOM, pp. 902--913 (2005)
8. Chen, B., Jamieson, K., Balakrishnan, H., Morris, R. Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. In: Proceedings of the ACM/IEEE international conference on Mobile Computing and Networking, July, (2001)
9. Xu, Y., Heidemann, J., Estrin, D. Geography-informed Energy Conservation for Ad-hoc Routing. In: Proceedings of the Seventh Annual ACM/IEEE International Conference on Mobile Computing and Networking, pp. 70--84 (2001)
10. W. Zhang, G. Cao. DCTC: Dynamic Convoy Tree-Based Collaboration for Mobile Target Tracking. IEEE Transactions on Wireless Communications, vol. 3, no. 5, 1689-1701 (2004)
11. Liu, J., Cheung, P., Guibas, L., Zhao, F.: A Dual-Space Approach to Tracking and Sensor Management in Wireless Sensor Networks. WSNA, pp. 131--139 (2002)
12. Duckham, M., Nittel, S., Worboys, M.: Monitoring dynamic spatial fields using responsive geosensor network. In: Proceedings of the 13th annual ACM international workshop on Geographic information systems, pp. 51--60 (2005)
13. Meng, X., Li, L., Nandagopal, T., Lu, S.: Event contour: an efficient and robust mechanism for tasks in sensor networks. Technical Report, UCLA (2004)
14. The Network Simulator – NS2, www.isi.edu/nsnam/ns