Distributed Homology Algorithm to Detect Topological Events via Wireless Sensor Networks

Christopher Farah, Frederick Schwaner, Ali Abedi, Michael Worboys

University of Maine, Orono, ME, 04469, USA

Abstract

Wireless sensor networks (WSNs) can span large geographical regions and collaboratively monitor environmental phenomena, e.g. forest fires. By designing a WSN to detect changes to such phenomena, current environmental monitoring systems could be supplemented, if not replaced. This research focuses on incremental insertion events, arising from the elevation of a single node's sensor reading between two consecutive states of network operation. Homology, a field of topology, is used to detect and differentiate between incremental insertion events of interest. Particular homology tools are translated to a distributed environment for 2-dimensional WSN deployments. The result is a novel distributed algorithm that can compute an incremental insertion event associated with a region comprising n nodes in $O(n)$ time, using $O(n)$ storage, and $O(n)$ data passed via messages. A small-scale, laboratory testbed is developed to evaluate the algorithm. Deployment results indicate that only nodes in physical proximity to an event are tasked, thereby conserving network resources and allowing multiple disparate events to be simultaneously monitored. Further, transmission cost is shown to vary linearly with the size of the evolving region, confirming one component of the formal analysis. [1]

Index Terms

Distributed algorithms, event detection, sensor systems and applications, spatiotemporal phenomena, topology, wireless sensor networks

I. Introduction

A wireless sensor network (WSN) is comprised of sensor nodes, tiny untethered computing devices, each equipped with one or more custom or off-the-shelf sensors. Over the past decade, prototype WSNs have been deployed to monitor avian habitats [1], [2], volcanoes [3], [4], coal mines [5], and water pipelines [6]. If properly designed, a WSN can be robust, self-organizing, scalable, and energy-efficient [7]. As sensor node costs decrease and WSN power management improves, large-scale, long-term WSN deployments

[1]This work is supported by the National Science Foundation, under grants IIS-0916219 and DGE 0504494 and by NASA, under grant NASA-EP-10-01-5404438. Author's addresses: C. Farah and M. Worboys, Department of Spatial Information Science and Engineering, University of Maine; A. Abedi and F. Schwaner, Department of Electrical and Computer Engineering, University of Maine.

become feasible, offering users unprecedented access to real-time, high resolution environmental data [8], [9].

The goal of this research is to enhance environmental monitoring and management by developing a distributed event detection algorithm for WSNs. Consider Figure 1, three chronological frames of the 1988 Yellowstone Forest Fires from August 16^{th} to September 16^{th} [10]. In the figure, regions are colored to distinguish between the progressions of independent fires, e.g., the dark brown region represents the spread of the North Fork Fire. The three frames reveal a number of changes the fires underwent, including appearance (of a new fire), merge (of two or more existing fires), accretion (of an existing fire), and self-merge (of an existing fire), or the merge of a fire with itself. Knowledge of these simple changes, along with location information, could prove valuable to emergency management teams. For example, the merging of two fires could block an existing evacuation route, thereby requiring a new evacuation route to be identified. With such applications in mind, this research focuses on detecting events like those in Figure 1. Attention is restricted to *incremental insertion events*, events arising from the elevation of a single node's sensor reading between two consecutive states of network operation. With this foundation, more complex events can be addressed.

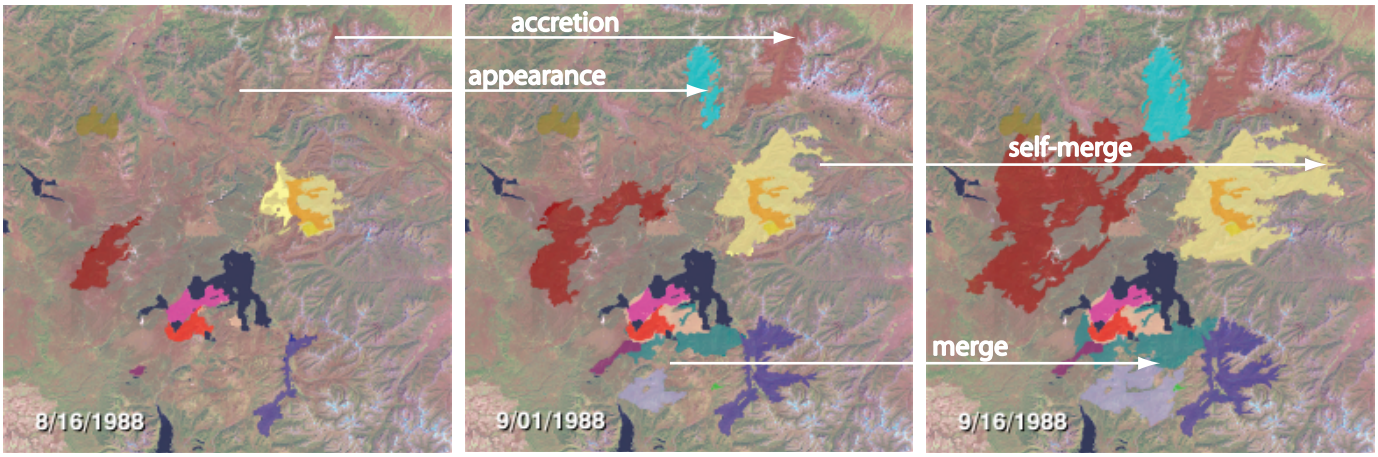

Fig. 1. Three frames of the Yellowstone forest fires of 1988, spanning one month, where colored regions distinguish between the progressions of independent fires. This research addresses some of the changes these fires underwent, as shown above.

Homology is a field of algebraic topology used to classify the connectivity and orientation of topological spaces. If two topological spaces are not homologically equivalent, then they cannot be topologically equivalent. In the context of the events in Figure 1, if two different states of (the evolving region corresponding to) a fire are not homologically equivalent, then the fire has undergone a topological change. Homology is able to differentiate between the events in Figure 1, thereby supporting event detection relevant to this research. Since algorithmic methods of homology have only been established for a centralized computing environment, one key step of this research is translating fundamental homology

concepts to a distributed environment.

As a surrogate for an environmental deployment or large-scale event detection, a laboratory testbed of a 2-dimensional, wirelessly connected sensor network is developed. Events are modeled as a sequence of bright lights delivered over the sensors. A distributed homology algorithm is developed to compute ongoing changes detected by the WSN. To evaluate the algorithm, the data transmitted per node per computed change are captured and analyzed to determine localization of computation and experimental communication complexity. In the remainder of the paper, we summarize related work and the fundamentals of homology that will be used in this research (Section II), develop and analyze the algorithm (Section III), outline the testbed set-up and present experimental results (Section IV), and conclude with a discussion of results, summary of findings, and future work (Section V).

II. Background

A. Related Work

This research lies at the intersection of topological event detection, distributed algorithms, WSNs, and homology. Given the challenges associated with embedding an algorithm into a WSN, ongoing research commonly addresses the development in stages. Jiang and Worboys formalized a framework for topological event detection, representing each stage of the space under investigation as a tree [11]. Between two states of a space, the corresponding tree representations infer one or more topological events. In later work, Jiang and Worboys embedded the model into a simulation environment and used node data to construct and update the tree representations in a distributed manner [12]. Funke used the hop count metric to detect holes and boundaries in a WSN's geometric distribution as a means of routing messages more efficiently [13]. In this research, each of one to four beacon nodes is responsible for collecting hop count data from the remaining nodes in the network. This data is used to construct approximations of isolines. These approximated isolines identify external and internal (holes) boundaries of the WSN. In follow-up work, hole detection is distributed using a rubber-banding procedure [14]. Liu et al. used a dual-space approach to detect half-planes, maintaining data by a topological line sweep [15]. Three distributed solutions were proposed in the discussion: limiting computational cost through clustering, localizing dual space data required by nodes, and incremental construction of dual-space. Specific to homology, a centralized algorithm was developed to characterize the coverage of a WSN [16]. The algorithm was tested in a simulation environment using open source software available through the Computational Homology Project (CHomP) [17].The authors discussed the need to develop a distributed solution. Research beginning with a distributed model include: Fang et al., with a distributed hole detection algorithm, using angle-of-arrival data of messages between nodes [18]; Sadeq and Duckham [19], with a tree representation method like that in the work of Jiang and Worboys.

The research presented in this paper provides a distributed event detection algorithm using WSNs. With the exception of [15], related research developments have been tested in simulation environments. Since simplifying assumptions of a simulation environment, e.g. radio propagation modeling, can lead to significantly different outcomes from a true deployment, the authors elect to evaluate the algorithm via testbed deployment [20], [21].

B. The Betti Numbers

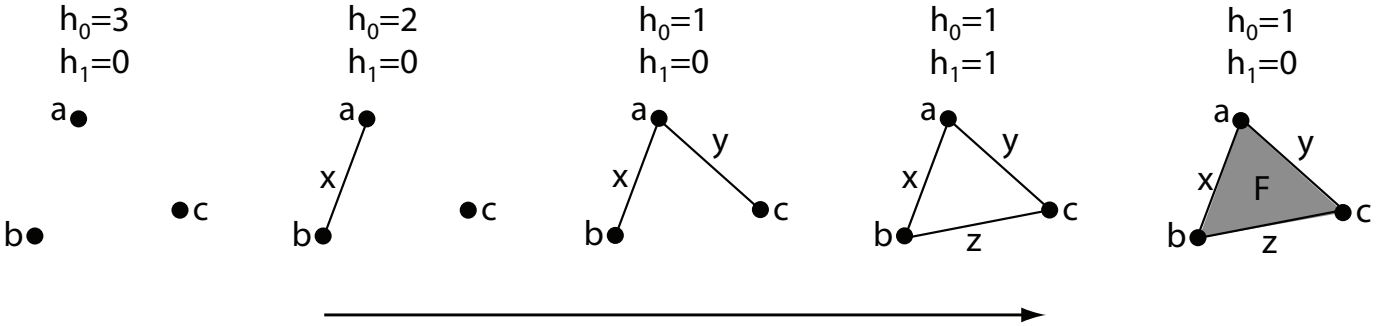

Fig. 2. Five stages of a simplicial complex as it evolves over time, and the corresponding 0^{th} and 1^{st} Betti numbers of each stage of the simplicial complex. As the figure indicates, h_0 and h_1 detect the number of components and holes of a simplicial complex, respectively.

A *topological invariant* is a property of a topological space that does not change when any *homeomorphism*, continuous mapping between two topological spaces with continuous inverse, is applied to that space [22]. The *Betti numbers* are topological invariants that measure the degree of connectivity of a topological space, including the number of connected components and the number of holes or genus [23]. Computation of the Betti numbers is machine tractable when the topological space under investigation is a *simplicial complex*, a collection of simplices, vertices, edges, faces, etc., in which the intersection of any two simplices is either empty or a simplex contained in the simplicial complex [24]. For $j > 0$, the j^{th} Betti number of a simplicial complex X can be obtained through the equation: $h_j(X) = S_j(X) - I_j(X) - I_{j-1}(X)$ [25]. Here, $S_j(X)$ is the rank of the basis of j-simplices of X and $I_j(X)$ is the rank of the incidence matrix between the j-simplices and (j-1)-simplices of X. When the topological space is understood or not required in the discussion, the j^{th} Betti number is notated h_j. Restricted to two dimensions, for each $j \geq 2$, $h_j(X)$ is identically zero [26]. The two non-trivial Betti numbers, $h_0(X)$ and $h_1(X)$, represent the number of connected components of X and the number of holes in X, respectively [27]. Figure 2 illustrates an evolving simplicial complex and the corresponding zeroth and first Betti numbers to clarify the role of each invariant. For a connected component, $h_0(X)$ is equal to one and therefore, the formula for $h_1(X)$ can be written [28]:

$$h_1(X) = -V(X) + E(X) - F(X) + 1 \tag{1}$$

Here, $S_0(X)$, $S_1(X)$, and $S_2(X)$ are written more simply as $V(X)$, $E(X)$, and $F(X)$, the ranks of the basis of vertices, edges, and faces, respectively. By substituting local node data for incidence matrices, nodes are alleviated of $O(n^2)$ storage and $O(n^3)$ time, for a WSN of n nodes [29]. (See Section III-D for analysis of storage and time complexity.) This substitution matches the goals of computational geometry, to replace computationally intensive operations with low-cost, intuitive operations when possible [30]. The j^{th} Betti number of a topological space at time t is notated $h_j(X, t)$.

III. Development

A. Linking Homology to the WSN

The *communication graph* is a graph in which vertices and edges correspond to sensor nodes and communication links between sensor nodes, respectively [31]. Data about the communication graph can be shared across nodes to support message routing. The Delaunay graph is a well-documented graph that has many good properties as a communication graph and is also a simplicial complex [32]. $K = (v(K), e(K), f(K))$ is the graph illustrated in Figure 3. Assuming the vertices and edges represent nodes and communication links, respectively, K is a communication graph and a Delaunay graph. Given this dual role, K can be used to route messages between sensor nodes, and K's homology can be computed, as well as the homology of subgraphs of K.

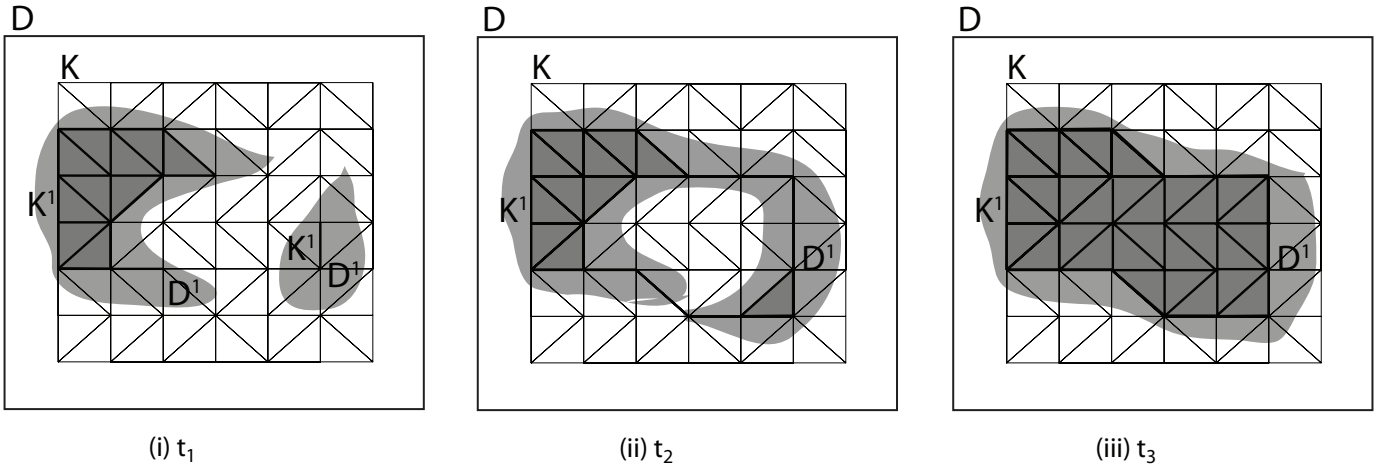

Fig. 3. Three stages of a WSN, where each node coincides with the point of intersection between two line segments. In the figure, D is the scalar field being monitored, D^1 is that part of the domain above sensor threshold (illustrated as grey regions), K is the communication graph of the WSN, and K^1 is that part of the communication graph induced by D^1.

A *(spatiotemporal) scalar field* is a spatiotemporal region in which every point has an associated scalar value [33]. Formally, it is a function s with spatiotemporal domain $D \times T$ and scalar codomain R such

that every point $(d,t) \in D \times T$ is assigned a unique scalar value $s(d,t) \in R$. This scalar value is tied to a *measurand*, a measurable physical property, such as light intensity, chemical concentration, or temperature. A Boolean scalar field is a scalar field in which the codomain is a two element set. In this work, a Boolean scalar field is established by setting a measurand threshold (e.g. 200^oF), mapping (d,t) to 1 if the scalar value at (d,t) exceeds the threshold and 0 otherwise. For each time t, D can be expressed as two disjoint sets, $D = D^0 \cup D^1$, where $D^i = \{d \in D \mid s\,(d) = i\}$. D^1, that portion of the scalar field above threshold, is comprised of zero or more connected components. These components (grey filled contours in Figure 3) and the changes they undergo are precisely what this work characterizes.

For each sensor node n in the set of nodes N, deployed in the spatial domain D, there exists a unique sensor value at each time t. We assume the sensor value and corresponding point value are the same, i.e., if the sensor's position coincides with $d_n \in D$, the sensor value at time t is $s(d_n,t)$. The set N can be separated analogously to the set D, i.e. $N = N^0 \cup N^1$ where $N^i = \{n \in N \mid s\,(d_n) = i\}$. Define $K^1 = (v(K^1), e(K^1), f(K^1))$ to be the induced graph that results when K's nodes are restricted to N^1 i.e. $v(K^1) = N^1$, $e(K^1) = \{(a,b) \in e(K) \mid a,b \in N^1\}$, and $f(K^1) = \{(a,b,c) \in f(K) \mid a,b,c \in N^1\}$. K^1 is a simplicial complex whose union comprises zero or more components and approximates D_1.

D, D^1, K, and K^1 are represented in Figure 3 at network times $t_1 < t_2 < t_3$, to illustrate the relationship between the scalar field and the Delaunay graph. Proceeding from left to right, the Betti numbers of K^1 and D^1 are: (i) $h_0(K^1,t_1) = 2$, $h_1(K^1,t_1) = 0$ and $h_0(D^1,t_1) = 2$, $h_1(D^1,t_1) = 0$; (ii) $h_0(K^1,t_2) = 1$, $h_1(K^1,t_2) = 1$ and $h_0(D^1,t_2) = 1$, $h_1(D^1,t_2) = 0$; and (iii) $h_0(K^1,t_3) = 1$, $h_1(K^1,t_3) = 0$ and $h_0(D^1,t_3) = 1$, $h_1(D^1,t_3) = 0$. Notice that the Betti numbers agree for D^1 and K^1 at times t_1 and t_3 but differ at time t_2: K^1 has a hole but D^1 does not. Such discrepancies arise from approximating a continuous scalar field with a finite scalar field. Identifying the appropriate level of resolution, i.e. density of sensor nodes, depends on the needs of the emergency management team interpreting the data. For example, if D^1 represents an evolving fire, then detecting the gap in D^1 at time t_2 is meaningful only if, e.g., the gap is broad enough to act as an escape route.

B. Detecting Events

Figure 4 illustrates incremental insertion events of interest. Each topological event involves a change to K^1, induced by D^1, which results in a change to one of the first two Betti numbers, Δh_0 and Δh_1. Here, Δ is used as in calculus, i.e. $\Delta h_i = h_i(X,t_2) - h_i(X,t_1)$. In each instance, Δh_0 and Δh_1 take on integer values. While appearance, hole loss, and accretion uniquely fix Δh_0 and Δh_1, merge and self-merge allow for a range of values, depending upon the number of distinct components that merge in the case of the former, and the number of "fingers" of a single component that merge in the case of the latter. Hence,

the inequalities in Figures 4(iii) and 4(iv). The inverse mapping for each event is an *incremental deletion event*. Incremental deletion events have been investigated in previous and ongoing work [11], [19], [34]. Detecting incremental deletion events through WSN testbed deployments is in progress.

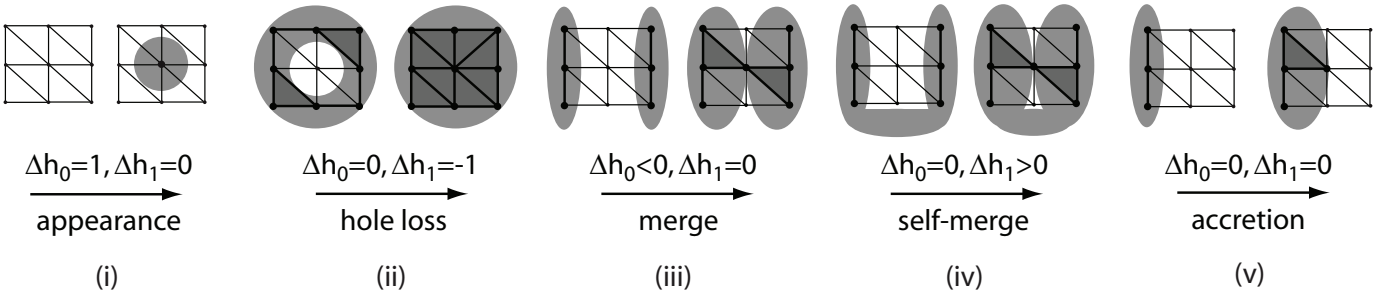

Fig. 4. The incremental insertion events investigated in this research in terms of changes to h_0 and h_1.

By Figure 4, computing Δh_0 and Δh_1 is sufficient to determine the incremental insertion event associated with an evolving component of K^1. While Δh_1 is explicitly computed from previous and current values of h_1, Δh_0 is determined by counting those components impacted by an incremental insertion event. The WSN does not maintain global values of h_0 and h_1 i.e. values that correspond to the total number of components and total number of holes associated with K^1.

Figure 5 presents the data elements required to compute h_0 and h_1. In frames (ii) and (iii), gray regions indicate elevated activity relative to sensor threshold. The *link* of a vertex n, notated $Lk(n) = (v(Lk(n)), e(Lk(n)))$, is the simplicial complex comprised of vertex n's neighbors, and edges between vertex n's neighbors. When the node in question is clear, notation simplifies to Lk. In Figure 5(i), the vertex and edge sets of node 1's link are $v(Lk) = \{2, 3, 4, 5, 6, 7\}$ and $e(Lk) = \{(2, 3), (3, 4), (4, 5), (5, 6), (6, 7)\}$, respectively. Referring to Figures 5(ii) and 5(iii), that part of the link induced by the region of elevated activity is notated Lk^1. Thus, the corresponding vertex and edge sets are $v(Lk^1) = \{3, 4, 5, 7\}$ and $e(Lk^1) = \{(3, 4), (4, 5)\}$, respectively. In words, $v(Lk)$ is the set of neighboring nodes, $e(Lk)$ is the set of neighboring communication links, $v(Lk^1)$ is the set of neighboring nodes above threshold, and $e(Lk^1)$ is the set of neighboring communication links where each node forming the link is in $v(Lk^1)$.

Each component of K^1 can be built up by adding link data incrementally. Between times t_1 and t_2, 2 components, 3 and 7, merge to form component 1. Thus $\Delta h_0 = 1 - 2 = -1 < 0$. To compute h_1, node 1 collects neighbor data, computes the values of V, E, F, and applies Formula 1. Since existing components and Lk^1 are mutually disjoint, the rank of the basis of k-simplices of the new component equals the sum of the ranks of the bases of k-simplices of contributing components and the rank of the basis of $(k-1)$-simplices of Lk^1. The values in this instance are computed as follows: $V(1) = V(3)+V(7)+1 = 3+1+1 = 5$, $E(1) = E(3) + E(7) + V(Lk^1) = 2 + 0 + 4 = 6$, and $F(1) = F(3) + F(7) + E(Lk^1) = 0 + 0 + 2 = 2$.

Thus, $h_1(1) = -V(1) + E(1) - F(1) + 1 = -5 + 6 - 2 + 1 = 0$ (confirming that there are no holes in component 1, as shown in Figure 5(iii)) and therefore, $\Delta h_1 = 0 - (0 + 0)$. Given the values of Δh_0 and Δh_1, the event occurring between t_1 and t_2 must be a merge. Using this process, the distributed code is developed.

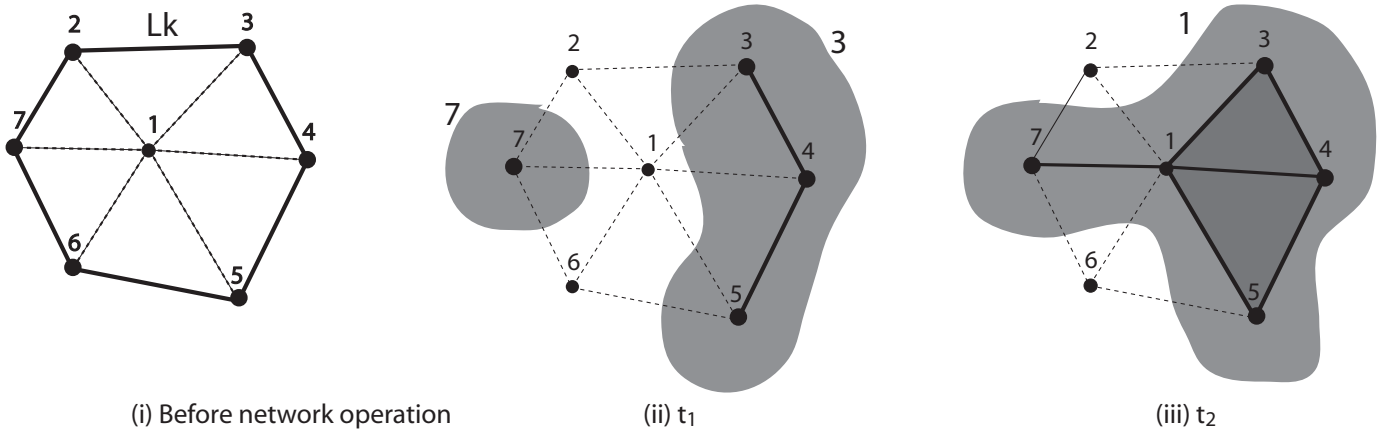

Fig. 5. Component 1 results from the merge of components 3 and 7 at node 1, between times t_1 and t_2. The link of node 1, Lk, is illustrated to better understand the supporting datasets, $v(Lk)$, $e(Lk)$, $v(Lk^1)$, and $e(Lk^1)$.

C. Algorithm

During an execution of the algorithm, each node can be in one of four states, INITIALIZE, WAIT, CRITICAL, and UPDATE. After INITIALIZE is complete, a node must be described as a wait node, a critical node, or an update node. Only one node can be in its CRITICAL state at any stage of an algorithm's execution. Three different events (interrupts) can occur to, or be scheduled by, each node: sample timer fires, receive message, and response timer fires. The first event indicates that the environment should be sampled via a designated sensor, the second event indicates that a message has been received, and the third event indicates that a message needs to be resent. There are four types of messages that can be sent: data request, acknowledge receipt of data request (ACK), node data, and update. The first message is transmitted by a node in its critical state; the second message, by a node in update or wait state; the third and fourth messages, by a node in update state. Node data and update contain the following data elements: component C, vertex count V, edge count E, face count F, the first Betti number $h_1(t_1)$, and Δh_1. Flow charts of the distributed algorithm are presented in Figures 6 and 7. Each individual process, required task or event, is numbered to ease reference. For example, the transmission of node data by a node in wait state (Figure 6 will be referred to as STATE:WAIT 2.1.

WAIT: A wait node sets its sample timer. When the sample timer fires, the node polls the environment through a designated sensor on its sensor board and enters CRITICAL state if sensor status has risen

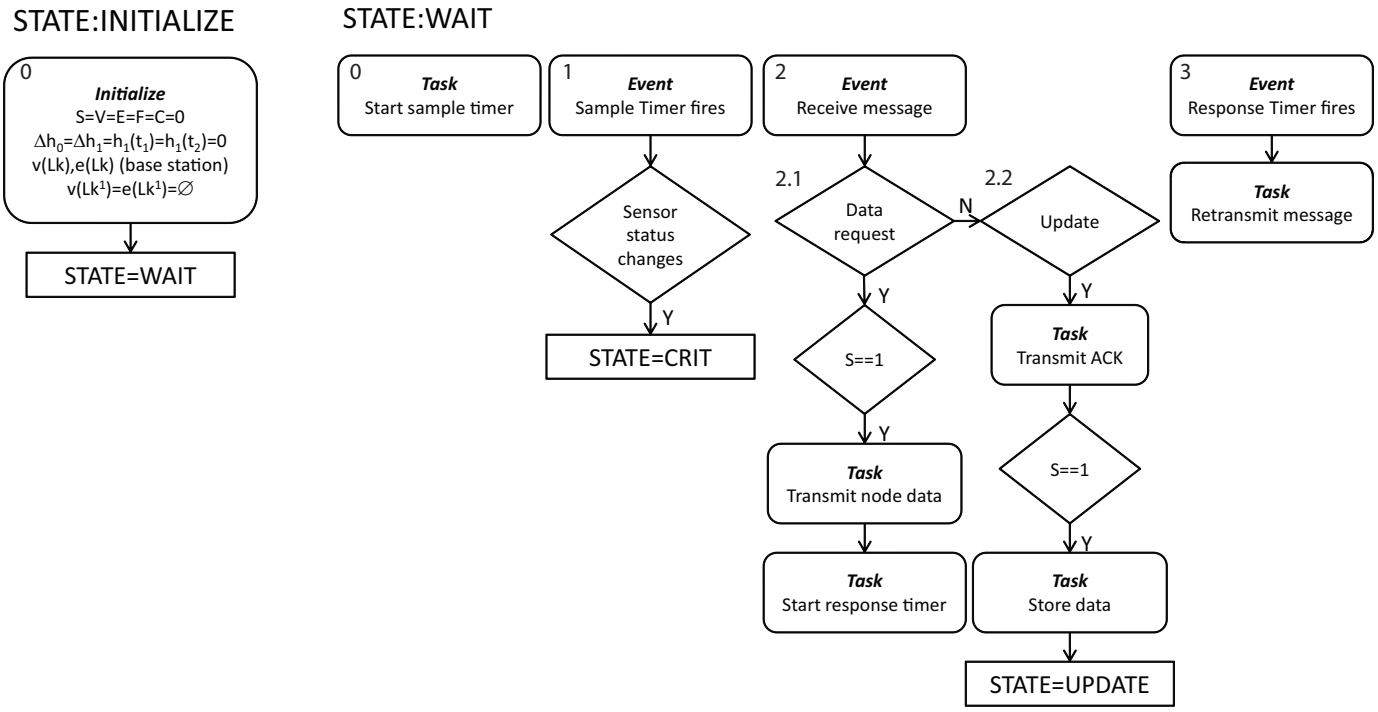

Fig. 6. Flow charts of the INITIALIZE and WAIT states of the algorithm, where each required task and event is numbered for clarity.

above threshold. If sensor status is below threshold, then the node resets its sample timer and remains in WAIT state. If a wait node receives a data request message, and if its sensor status relative to threshold is high, then it transmits a node data message to the critical node and sets a response timer. If the response timer fires, this message is resent. If a wait node receives an update message, it transmits and ACK and stores the data if its sensor status relative to threshold is high.

CRITICAL: The critical node transmits a data request to each neighbor. As neighbors respond, the critical node builds $v(Lk^1)$ and $e(Lk^1)$ from $v(Lk)$ and $e(Lk)$. For each unique existing component contributing to the new component, V, E, and F are updated appropriately. After computing h_1, Δh_0, and Δh_1, as illustrated in the previous section, the critical node proceeds to UPDATE state.

UPDATE: An update node starts a response timer and broadcasts an update message to each of its neighbors. If an ACK is not received for a particular recipient node, the message is resent. If an update message is received, the node ACKs the transmitting node. After the update node has received ACKs from all of its neighbors, the node proceeds to WAIT state.

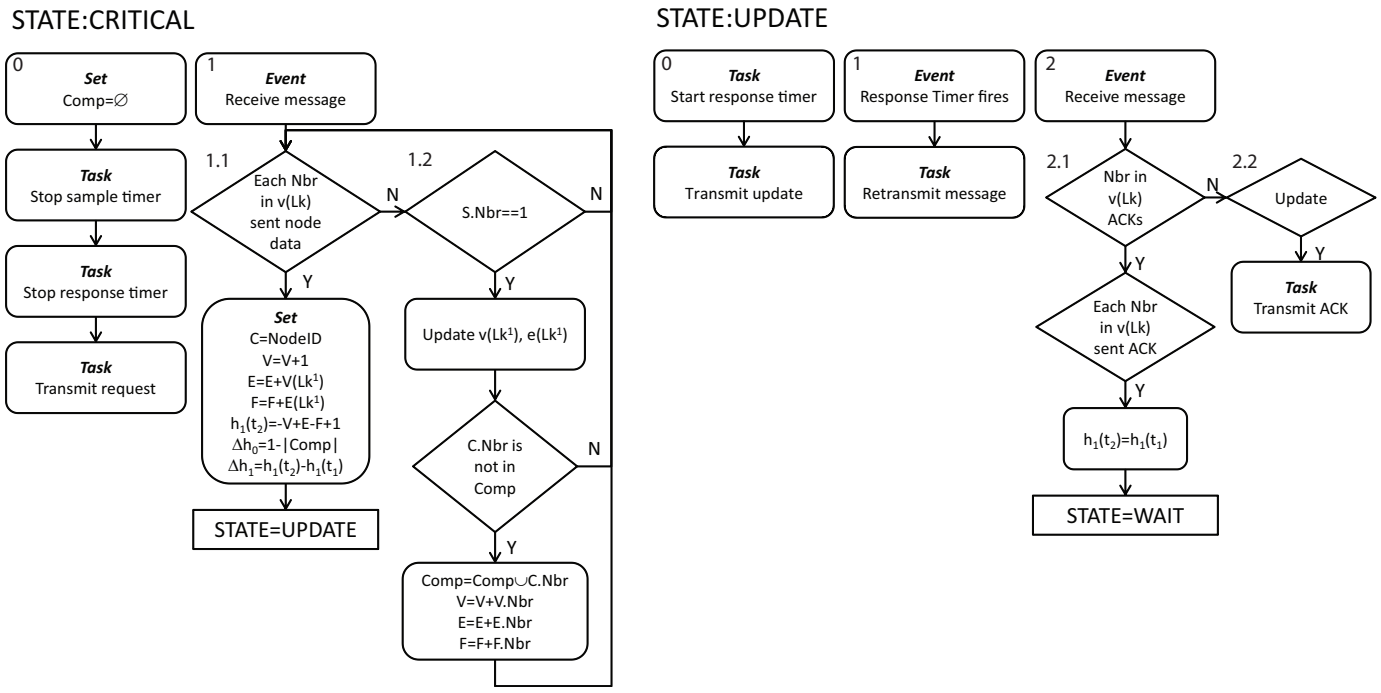

Fig. 7. Flow charts of CRITICAL and UPDATE states of the algorithm. The numbering convention is the same as in Figure 6.

D. Analysis

This section analyzes the algorithm to determine the worst case time complexity, storage complexity, and message complexity. A key assumption is that the degree of the deployed WSN's communication graph, Delaunay graph, is bounded by some integer k, independent of the number of nodes in the WSN, n. This assumption is reasonable for grid deployments, or randomized deployments in which a pathological geometric distribution occurs with low likelihood [32].

Theorem: The algorithm computes an incremental insertion event for a component comprised of n nodes in $O(n)$ time.

Proof: Let k be the maximum degree of the communication graph of the WSN. Steps STATE:CRITICAL 1.2 and STATE:UPDATE 2.1 comprise the highest time complexity, since the node in question must evaluate whether a neighbor has transmitted a node data or ACK message. Using a standard quadratic time search, this process is $O(k^2)$ in each instance. Since only n nodes have sensor reading above threshold, $(n-1)$ nodes will execute STATE:UPDATE 2.1 and the critical node will execute steps STATE:CRITICAL 1.2 and STATE:UPDATE 2.1. In other words, each of n nodes requires constant time to participate in the computation. The result follows. ■

Theorem: The algorithm computes an incremental insertion event for a component comprised of n nodes using $O(n)$ storage.

Proof: Let M be a constant bound for simplicial complex data, component membership data, topology

data, messages en queue, and supporting data (operating system, source code, local address data) stored at each node; let k be the maximum degree of the WSN's communication graph. Clearly, an upper bound for the total storage required is Mkn. The result follows. ■

Theorem: The algorithm computes an incremental insertion event for a component comprised of n nodes using $O(n)$ transceived data.

Proof: There are five mutually exclusive node types that potentially contribute to the total message complexity, associated with computations: the critical node $Crit$, the critical node's neighbors above threshold $v(Lk^1)$, non-neighbors of the critical node that belong to the evolving component $v(C) - v(Lk^1) - Crit$, neighbors (hence below threshold) of the evolving component $Nbr(C)$, and non-neighbors of the evolving component $v(K) - Nbr(C) - v(C)$, where Nbr notates nodes that neighbor the node(s) in question. Let the maximum number of neighbors for any node in the network be k, total transmitted data be Tx, total received data be Rx, the cost of a data request or ACK message be m_1, and the cost of a node data or update message be m_2.

For each node, data transmission occurs if STATE:WAIT 2.1, STATE:WAIT 2.2, STATE:WAIT 3, STATE:CRITICAL 0, STATE:UPDATE 0, STATE:UPDATE 1, or STATE:UPDATE 2.2 is executed. Assuming retransmission only contributes constant cost to total transmission (and consequently, to total reception), STATE:WAIT 3 and STATE:UPDATE 1 are excluded from the analysis. Bounds for transmission costs per node type are:

$Tx(Crit) \leq 2m_1k + m_2k$ (Cost of STATE:CRITICAL 0, STATE:UPDATE 0, and STATE:UPDATE 2.2)
$Tx(v(Lk^1)) \leq m_1(k+1) + m_2(k+1)$ (Cost of STATE:WAIT 2.1, STATE:WAIT 2.2, STATE:UPDATE 0, and STATE:UPDATE 2.2)
$Tx(v(C) - v(Lk^1) - Crit) \leq m_1(k+1) + m_2k$ (Cost of STATE:WAIT 2.2, STATE:UPDATE 0, and STATE:UPDATE 2.2)
$Tx(Nbr(C)) \leq m_2k$ (Cost of STATE:WAIT 2.2)
$Tx(v(K) - Nbr(C) - v(C)) = 0$

Since there is one critical node, a maximum of k neighbors above threshold, $n-k-1$ nodes remaining in the component, and an upper bound of kn for nodes neighboring the component, $Tx \leq 2m_1k + m_2k + [m_1(k+1) + m_2(k+1)]k + [m_1(k+1) + m_2k](n-k-1) + m_2k^2n$, which reduces to:
$Tx \leq (m_1k + m_1 + m_2k^2 + m_2k)n + m_1k + m_2k - m_1$

In a similar fashion, the total received data during the computation is bounded by:
$Rx \leq (m_1k^2 + m_1k + m_2k + m_2)n + m_1k - m_2$

Since total message complexity is proportional to $Tx + Rx$, and m_1, m_2, and k are constants, the result follows. ■

Thus, worst case time, storage, and message complexity are $O(n)$ for an execution of the algorithm involving one incremental insertion to a component comprised of n nodes. Message complexity outcomes are partially confirmed by testbed results in Section IV-B.

IV. Experiment

A. Testbed Set-up

Crossbow's TelosB mote is an open source platform designed to enable and accelerate WSN experimentation and understanding. TelosB is IEEE 802.15.4/ZigBee compliant, has a data rate of 250 kbps, an 8 MHz TI MSP430 microcontroller with 10kB RAM, and 1MB external flash for data logging [35]. The TelosB motes in this experiment are equipped with light sensors and run TinyOS 1.1.10, an open source, component-based, energy-efficient operating system developed by UC Berkeley which supports large scale, self-configuring sensor networks [36].

An experimental testbed of TelosB motes is developed at the Wireless Sensor Network Laboratory (WiSe-Net) at the University of Maine Department of Electrical and Computer Engineering [37]. The primary goals of the testbed are to validate the algorithm, confirm localization of computation, and determine the relation between the communication cost and component size of topological events being computed. Since transception draws more power than internal processing, and transmission and reception costs are roughly equivalent for the TelosB, overall network cost is assessed through data transmission per node, in bytes [36]. Deployments consist of one 9-node deployment and two 16-node deployments. Nodes are configured in square grid and randomized geometries. Over the 3 WSN deployments, a total of 50 event trials are conducted. For each event trial, a sensor node is exposed to light, either creating a new component or altering an existing component, thereby simulating one of the five topological events of interest. Groups of event trials are executed sequentially to better simulate evolving environmental phenomena and to confirm the algorithm operates correctly over time. Prior to each deployment, MATLAB R2009b is used to determine precise node locations, generate the Delaunay graph of the WSN, and parse this data to node-level Delaunay information [38]. Code corresponding to the second task makes use of algorithms available through the open source Computational Geometry Algorithms Library [39].

Twelve consecutive event trials for the 16-node, grid deployment are shown in Figure 8. Results for the remaining 50 trials prove similar and are not presented for space reasons. Filled white circles

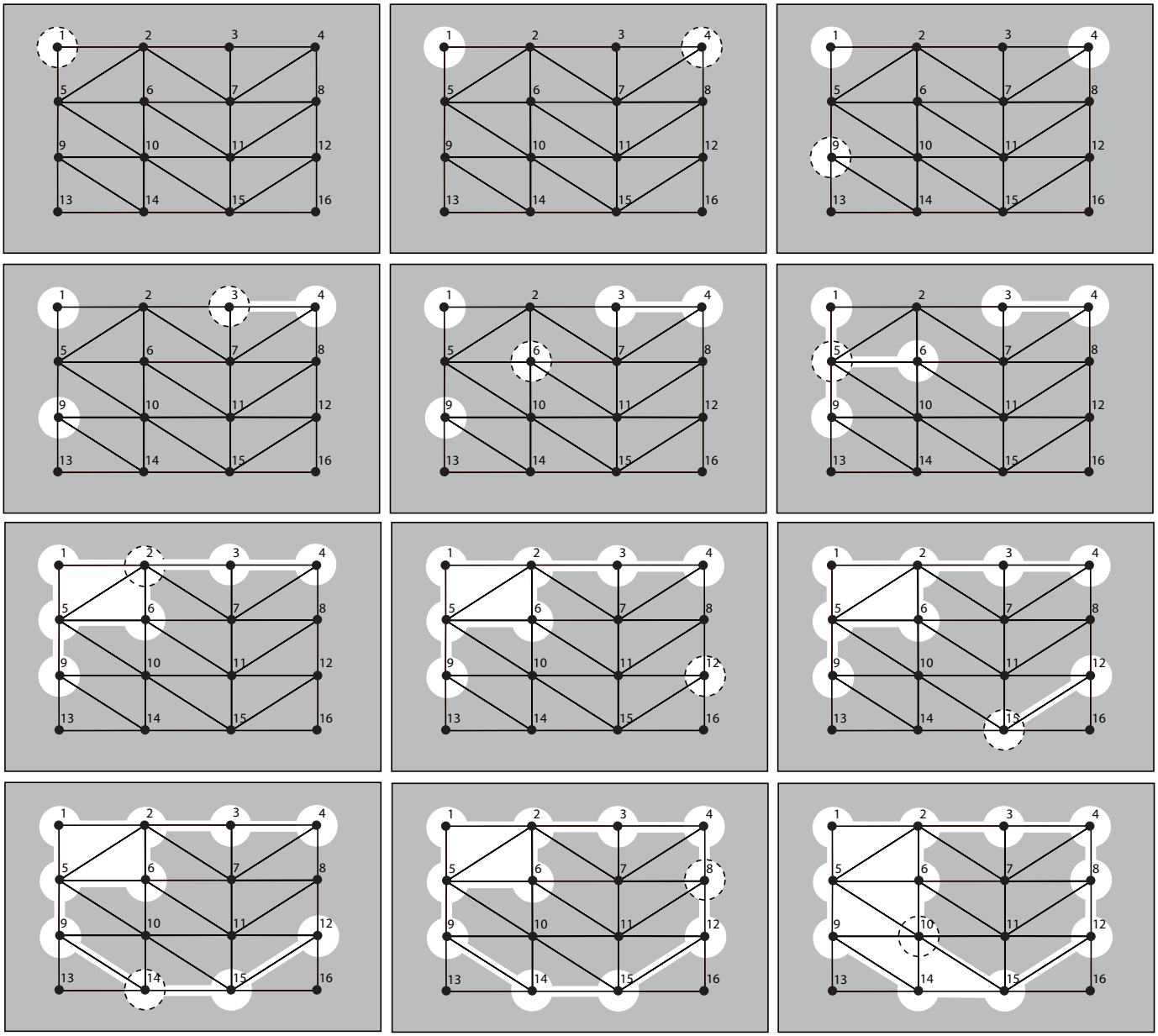

Fig. 8. Twelve consecutive event trials for a 16-node testbed deployment of TelosB motes, ordered from left-to-right, top-to-bottom. A white circle with dashed circumference indicates that light has been delivered to the node shown for the current event trial. All other circles indicate the light was delivered during a previous event trial. White edges/faces do not represent delivery of light to the WSN but the construction of 1- and 2-simplices by the algorithm.

illustrate that a sequence of bursts of light has been delivered to nodes using a halogen lamp. A dashed circumference indicates that light has been delivered to the node shown for the current event trial, i.e. a dashed circumference marks the critical node. All other circles indicate the light was delivered during a previous event trial. White edges/faces do not represent delivery of light to the WSN but the construction of 1- and 2-simplices by the algorithm. For this figure, frames are in chronological sequence from left-to-right column, top-to bottom row.

B. Results

Figure 9 presents the corresponding number of bytes transmitted per node for each of the twelve event trials in Figure 8, in the same order, e.g. the top left frame of Figure 8 corresponds to the top left frame of Figure 9. Comparing the two figures, a few observations can be made. First, the critical node and nodes belonging to the evolving component transmit the most data. Second, neighbors of the evolving component that are below threshold transmit very little data; generally an ACK package. Third, nodes that are two or more hops from the evolving component, including nodes belonging to a different component, transmit no data during computations. Thus, each event draws on nodes that are spatially tied to the event.

To quantify the aforementioned observations, mean data transmitted for each category of node over the 50 event trials is shown in Figure 10(i). The critical node, $Crit$, averages 54 bytes per event trial; neighbors of the critical node above sensor threshold, $v(Lk^1)$, average 59 bytes per event trial; nodes belonging to the same component C as the critical node, $v(C) - v(Lk^1) - Crit$, average 54 bytes per event trial; nodes neighboring component C, $Nbr(C)$, average 9 bytes per event trial; nodes not neighboring component C, $v(K) - v(C) - Nbr(C)$, transmit precisely 0 bytes per event trial.

As shown in Figure 10(ii), transmitted data, Tx (bytes) and component size, C (number of nodes), display a linear relationship, $Tx = 57.72C + 3.50$, with component size accounting for about 97% of the variance of Tx ($R^2 = 0.9735$). Analysis of the source data indicates that the remaining variance of Tx is explained by: number of critical node's neighbors and data retransmission arising from lost packets.

V. Discussion of Results and Conclusions

Testbed results confirm four important features of the distributed algorithm:

1) h_1, Δh_0, and Δh_1 are correctly computed in-network for each of the events investigated.
2) Local data is sufficient to maintain and update component membership for each node in the network.
3) The total data transmitted while computing an event increases linearly with component size.
4) Nodes more than 1 hop from the component undergoing change do not participate in the computation.

This research is restricted to incremental change, that is, a single node changes sensor status between two consecutive network states. By incorporating a network token, correct computations could be executed for non-incremental changes as well [40]. In particular, a unique network token could prevent the occurrence of multiple critical nodes at any given state of operation, ensuring non-incremental changes could be computed as a sequence of incremental changes. To accommodate large-scale, non-incremental change, scalability could be improved through WSN clustering [41]. For example, by assigning cluster heads to groups of nodes in geographical proximity, updates could be computed and propagated via cluster heads as opposed to the flooding routine used in this research. Incorporating clustering would result in a trade-off

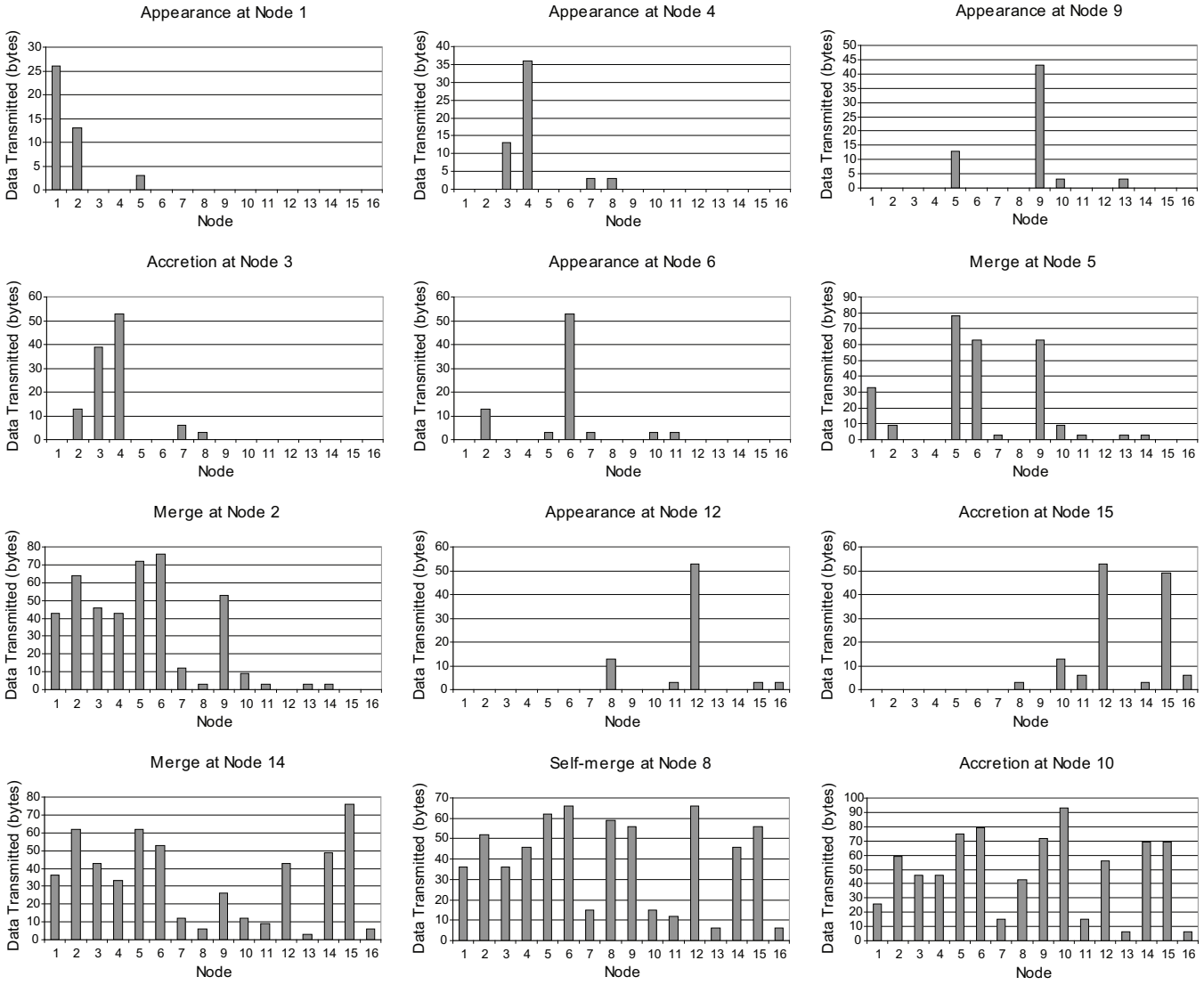

Fig. 9. Total transmitted data per node for each event trial shown in Figure 8.

between the cost/time of flat routing and the resources required to maintain data about the entire cluster at the cluster head. Finally, a more generalized notion of the link, Lk, would have to be developed to account for the link of an entire cluster, versus individual nodes, as in this work.

While node positions are determined prior to deployment, virtual coordinate algorithms have been, and continue to be, developed that allow for on-the-fly calculation of the relative coordinates of the nodes in a network [42], [43]. Furthermore, while the Delaunay graph can only be computed in a centralized fashion, there are variants of the Delaunay graph that can be computed in a distributed fashion [44]. Incremental algorithms have been developed to update a Delaunay graph when a limited number of nodes are added to, or removed from the graph, or when nodes are not stationary [45], [46]. Therefore, in future work/deployments, a distributed virtual coordinate algorithm could be used to deliver each node its

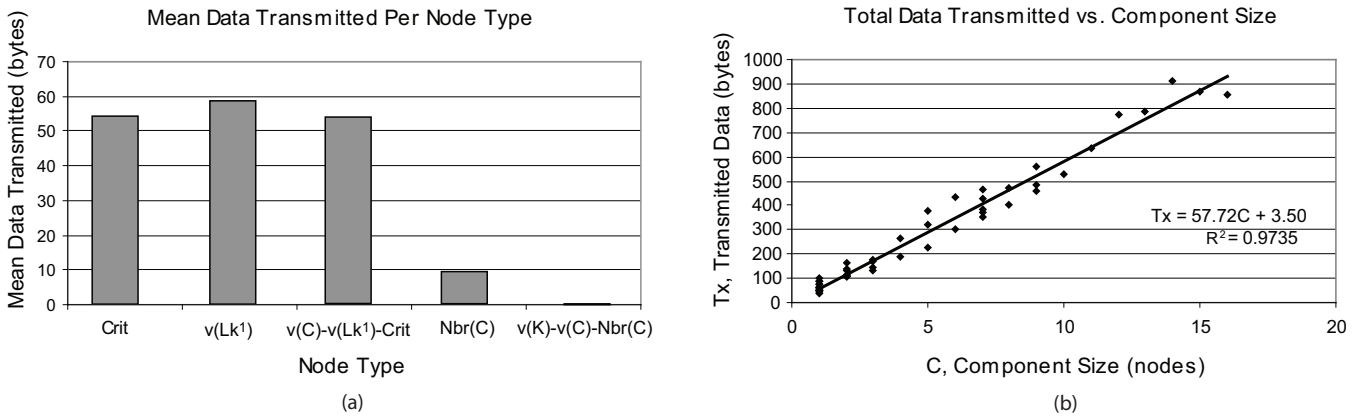

Fig. 10. For the 50 event trials, (i) mean data transmitted over all events per node type, and (ii) linear regression of total data transmitted against component size

relative coordinates and nodes could compute and store Delaunay-like neighbor information after relative coordinates are determined, and also update this data when nodes are added to, or removed from, the WSN.

The algorithm developed in this research only has one mechanism to deal with failed communication, message retransmission. To extend this work, more significant communication failure, along with node failure, need to be considered. For example, how should computations proceed if a critical node discovers one of its neighbors is no longer transmitting messages? Such failure could occur when the critical node is computing the simplicial complex of the evolving component, along with the component's topology (STATE:CRITICAL 1 of the algorithm) or when the critical node has completed computations, transmitted updates, and is awaiting ACKs from its neighbors (STATE:UPDATE 2 of the algorithm) before entering WAIT state. Persisting node failure would require $v(Lk)$ and $e(Lk)$ to be recomputed for all nodes neighboring the failed node, and simplicial complex and topology data to be updated appropriately.

Incremental deletion events [34], and 3-dimensional deployments are not addressed in this research. A foundation to detect incremental deletion events, when combined with a foundation to detect incremental insertion events, provide a richer, more realistic framework to monitor environmental phenomena, since one would not expect monotone behavior in terms of sensor status relative to threshold. More likely, a node's sensor value will rise above and dip below sensor threshold multiple times during a monitoring scenario, particularly for long-term deployments. 3-dimensional deployments can potentially identify topological phenomena that do not exist in 2-dimensions, e.g., the formation of a cavity in an evolving component. If monitoring a toxic gas plume, being able to detect and distinguish between a cavity and a tunnel of non-toxic air could equate to identifying and differentiating between a temporary safe zone and an evacuation route.

This paper presents a novel distributed homology algorithm to detect topological events using WSNs. The approach is incremental and draws on local communication graph data stored at each node to correctly compute changes to the first two Betti numbers, i.e. Δh_0 and Δh_1. A testbed of TelosB motes was implemented to validate the algorithm. The algorithm demonstrated linear message complexity with the size of the phenomena being monitored. Furthermore, localized, autonomous computation was demonstrated: nodes associated with all other components or two hops from the component in question did not participate in computations. Ongoing work investigates incremental deletion events, non-incremental events, and 3-dimensional, non-planar deployments.

REFERENCES

[1] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, ser. WSNA '02. New York, NY, USA: ACM, 2002, pp. 88–97. [Online]. Available: http://doi.acm.org/10.1145/570738.570751

[2] R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson, and D. Culler, "An analysis of a large scale habitat monitoring application," in *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*. New York, NY, USA: ACM, 2004, pp. 214–226.

[3] W.-Z. Song, R. Huang, M. Xu, A. Ma, B. Shirazi, and R. LaHusen, "Air-dropped sensor network for real-time high-fidelity volcano monitoring," in *MobiSys '09: Proceedings of the 7th international conference on Mobile systems, applications, and services*. New York, NY, USA: ACM, 2009, pp. 305–318.

[4] G. Werner-Allen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh, "Fidelity and yield in a volcano monitoring sensor network," in *OSDI '06: Proceedings of the 7th symposium on Operating systems design and implementation*. Berkeley, CA, USA: USENIX Association, 2006, pp. 381–396.

[5] M. Li and Y. Liu, "Underground structure monitoring with wireless sensor networks," in *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*. New York, NY, USA: ACM, 2007, pp. 69–78.

[6] I. Stoianov, L. Nachman, S. Madden, and T. Tokmouline, "Pipeneta wireless sensor network for pipeline monitoring," in *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*. New York, NY, USA: ACM, 2007, pp. 264–273.

[7] D. Gay, M. Welsh, P. Levis, E. Brewer, R. V. Behren, and D. Culler, "The nesc language: A holistic approach to networked embedded systems," 2003, pp. 1–11.

[8] D. Gay, P. Levis, and D. Culler, "Software design patterns for tinyos," *Transactions on Embedded Computing Systems*, vol. 6, no. 4, p. 22, 2007.

[9] B. Shen and A. Abedi, "A simple error correction scheme for performance improvement of ieee 802.15.4," in *ICWN '07: International Conference on Wireless Networks*, 2007.

[10] "Wildfire growth around yellowstone national park in 1988, scientific visualization studio, goddard space flight center," 2010, http://svs.gsfc.nasa.gov/vis/.

[11] J. Jiang and M. Worboys, "Event-based topology for dynamic planar areal objects," in *International Journal of Geographic Information Science*, 2007.

[12] ——, "Detecting basic topological changes in sensor networks by local aggregation," in 16^{th} *ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2008.

[13] S. Funke, "Topological hole detection in wireless sensor networks and its applications," in *DIALM-POMC '05: Proceedings of the 2005 Joint Workshop on Foundations of Mobile Computing*. New York, NY, USA: ACM, 2005, pp. 44–53.

[14] S. Funke and N. Milosavljevic, "Network sketching or: "how much geometry hides in connectivity? - part ii"," in *Technical Report: Max Planck Center for Visual Computing and Communication*. Max-Planck-Institut für Informatik,, 2007, pp. 1–10.

[15] J. Liu, P. Cheung, F. Zhao, and L. Guibas, "A dual-space approach to tracking and sensor management in wireless sensor networks," in *WSNA '02: Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*. New York, NY, USA: ACM, 2002, pp. 131–139.

[16] R. Ghrist and A. Muhammad, "Coverage and hole-detection in sensor networks via homology," in *IPSN '05: Proceedings of the 4th International Symposium on Information Processing in Sensor Networks*. Piscataway, NJ, USA: IEEE Press, 2005, p. 34.

[17] "Computational homology project," 2010, http://chomp.rutgers.edu/.

[18] Q. Fang, J. Gao, and L. Guibas, "Locating and bypassing holes in sensor networks," *Mobile Networking Applications*, vol. 11, no. 2, pp. 187–200, 2006.

[19] M. Sadeq and M. Duckham, "Effect of neighborhood on in-network processing in sensor networks," *Proceedings of the 8th International GIScience Conference*, 2008.

[20] D. Kotz, C. Newport, R. S. Gray, J. Liu, Y. Yuan, and C. Elliott, "Experimental evaluation of wireless simulation assumptions," in *Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*, ser. MSWiM '04. New York, NY, USA: ACM, 2004, pp. 78–82. [Online]. Available: http://doi.acm.org/10.1145/1023663.1023679

[21] L. Bergamini, C. Crociani, A. Vitaletti, and M. Nati, "Validation of wsn simulators through a comparison with a real testbed," in *Proceedings of the 7th ACM workshop on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*, ser. PE-WASUN '10. New York, NY, USA: ACM, 2010, pp. 103–104. [Online]. Available: http://doi.acm.org/10.1145/1868589.1868611

[22] J. Munkres, *Topology (2nd Edition)*. Prentice Hall, 2000.

[23] M. Henle, *A Combinatorial Introduction to Topology*. Dover Publications, March 1994.

[24] J. Munkres, *Elements of Algebraic Topology*. Westview Press, December 1993.

[25] J. Vick, *Homology Theory: An Introduction to Algebraic Topology*, 2nd ed. New York, NY, USA: Springer-Verlag, Inc., 1994.

[26] M. Agoston, *Algebraic Topology: A First Course*. Marcel Dekker, Inc., 1976.

[27] A. Fomenko, *Visual Geometry and Topology*. Springer-Verlag, 1994.

[28] A. Wallace, *Algebraic Topology: Homology and Cohomology*. New York, NY, USA: W.A. Benjamin, Inc., 1970.

[29] L. Kronsjö, *Algorithms: their complexity and efficiency (2nd ed.)*. New York, NY, USA: John Wiley & Sons, Inc., 1987.

[30] B. Chazelle, "Computational geometry: a retrospective," in *STOC '94: Proceedings of the 26th Annual ACM Symposium on Theory of Computing*. New York, NY, USA: ACM, 1994, pp. 75–94.

[31] F. Zhao and L. Guibas, *Wireless Sensor Networks: An Information Processing Approach*. San Francisco, CA, USA: Morgan Kaufmann Publications, Elsevier, Inc., 2004.

[32] H. Edelsbrunner, M. J. Ablowitz, S. H. Davis, E. J. Hinch, A. Iserles, J. Ockendon, and P. J. Olver, *Geometry and Topology for Mesh Generation (Cambridge Monographs on Applied and Computational Mathematics)*. New York, NY, USA: Cambridge University Press, 2006.

[33] M. Duckham, S. Nittel, and M. Worboys, "Monitoring dynamic spatial fields using responsive geosensor networks," in *GIS '05: Proceedings of the 13th Annual ACM International Workshop on Geographic Information Systems*. New York, NY, USA: ACM, 2005, pp. 51–60.

[34] C. Farah, C. Zhong, M. Worboys, and S. Nittel, "Detecting topological change using a wireless sensor network," in *GIScience '08: Proceedings of the 5th international conference on Geographic Information Science*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 55–69.

[35] "Telosb mote platform," 2010, http://www.memsic.com/.

[36] P. Levis, "Tinyos: An open platform for wireless sensor networks," Stanford University, Nara, Japan, Tech. Rep., 2006.

[37] "Wise-net lab," 2010, http://wisenet.eece.maine.edu/.

[38] "Matlab and simulink," 2010, http://www.mathworks.com/.

[39] "Computational geometry algorithms library," 2010, http://www.cgal.org/.

[40] N. Lynch, "Distributed computing theory: algorithms, impossibility results, models, and proofs," in *STOC '07: Proceedings of the 39th Annual ACM Symposium on Theory of Computing*. New York, NY, USA: ACM, 2007, pp. 247–247.

[41] E. Belding-Royer, "Multi-level hierarchies for scalable ad hoc routing," in *Wireless Networks*. Netherlands: Kluwer Academic Publishers, 2003, pp. 461–478.

[42] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, "Vivaldi: A decentralized network coordinate system," in *Proceedings of the ACM SIGCOMM '04 Conference*, Portland, Oregon, August 2004.

[43] T. Moscibroda, R. O'Dell, M. Wattenhofer, and R. Wattenhofer, "Virtual coordinates for ad hoc and sensor networks," in *DIALM-POMC '04: Proceedings of the 2004 joint workshop on Foundations of mobile computing*. New York, NY, USA: ACM, 2004, pp. 8–16.

[44] J. Gao, L. Guibas, J. Hershberger, L. Zhang, and A. Zhu, "Geometric spanner for routing in mobile networks," in *MobiHoc '01: Proceedings of the 2nd ACM International Symposium on Mobile Ad hoc Networking & Computing*. New York, NY, USA: ACM, 2001, pp. 45–55.

[45] O. Devillers, "On deletion in delaunay triangulations," in *SCG '99: Proceedings of the 15th Annual Symposium on Computational Geometry*. New York, NY, USA: ACM, 1999, pp. 181–188.

[46] L. Guibas and D. Russel, "An empirical comparison of techniques for updating delaunay triangulations," in *SCG '04: Proceedings of the 20th Annual Symposium on Computational Geometry*. New York, NY, USA: ACM, 2004, pp. 170–179.