

Destination-based Cut Detection in Wireless Sensor Networks

Myounggyu Won and Radu Stoleru

Department of Computer Science and Engineering, Texas A&M University
 {mgwon, stoleru}@cse.tamu.edu

Abstract—Wireless Sensor Networks (WSNs) often suffer from disrupted connectivity caused by its numerous aspects such as limited battery power of a node and unattended operation vulnerable to hostile tampering. The disruption of connectivity, often referred to as *network cut*, leads to ill-informed routing decisions, data loss, and waste of energy. A number of protocols have been proposed to efficiently detect network cuts; they focus solely on a cut that disconnects nodes from the *base station*. However, a cut detection scheme is truly useful when a cut is defined with respect to multiple destinations (i.e., target nodes), rather than a single base station. Thus, we extend the existing notion of cut detection, and propose an algorithm that enables sensor nodes to autonomously monitor the connectivity to multiple target nodes. We introduce a novel reactive cut detection solution, the *Point-to-Point Cut Detection*, where given any pair of source and destination, a source is able to locally determine whether the destination is reachable or not. Furthermore, we propose a light-weight proactive cut detection algorithm specifically designed for a small set of target destinations. We prove the effectiveness of the proposed algorithms through extensive simulations.

Keywords—Wireless sensor networks; cut detection; energy efficiency

I. INTRODUCTION

Wireless sensor networks (WSNs), consisting of large numbers of low-cost and low-power wireless nodes, have recently been employed in many applications: disaster response [1], military surveillance [2], and medical care [3] among others. The inherent nature of WSNs such as unattended operation, battery-powered nodes, and harsh environments pose major challenges. One of the challenges is to ensure that the network is connected. The connectivity of the network can easily be disrupted due to unpredictable wireless channels, early depletion of node's energy, and physical tampering by hostile users. Network disconnection, typically referred as a *network cut*, may cause a number of problems. For example, ill-informed decisions to route data to a node located in a disconnected segment of the network might lead to data loss, wasted power consumption, and congestion around the network cut.

Several centralized algorithms have been proposed to efficiently detect a cut [4][5][6][7]. These algorithms attempt to detect a cut by assigning the task of network connectivity monitoring to a subset of nodes. In particular, Shrivastava et al. [7] proposed an algorithm to detect a linear cut in a WSN, by strategically deploying specially designated nodes, called sentinels. Some researchers have recently proposed distributed cut detection algorithms for WSNs [8][9]. In these schemes, each sensor node is able to autonomously determine the existence of a cut. A common aspect of existing cut detection algorithms is that they focus on a “binary problem”: is there a

cut in the network, or not? However, this may not be sufficient since, in some applications, despite the existence of a cut somewhere in the network, a sender can still communicate with a target node, if they are not disconnected by the cut. For example, some WSN applications adopt the strategy to deploy multiple sink nodes, in order to improve throughput and prolong network lifetime [10][11]. In these applications, detecting a cut to one sink node does not necessarily mean that a node in the disconnected network segment should refrain from reporting data, since the node may send its data to other connected sink nodes.

In this paper, we propose solutions for a more general cut detection problem – the *destination-based cut detection* problem. Unlike the traditional cut detection problem, we attempt to find a network cut between a sender and any node in a set of given destinations. We first propose Point-to-Point Cut Detection protocol (P2P-CD). P2P-CD allows a source node to identify a cut with respect to any destination node. In this protocol, the boundary of a cut is compactly represented as a set of linear segments. The compact representation of a cut allows the information on existing cuts (i.e., the shape and location of the cut) to be efficiently distributed throughout the network with small overheads. A source node, using the distributed information, locally determines whether any potential destination is reachable.

P2P-CD is a reactive algorithm; in other words, a cut is reactively detected in contrast to the proactive solutions that periodically probe the network for potential cuts; thus, P2P-CD is energy efficient. However, P2P-CD faces several challenges: each node has to store a data structure that contains the information on the cuts in the network, and nodes must be localized. Thus, we propose a light-weight cut detection algorithm (RE-CDM) particularly designed for the scenario, where nodes need to detect a cut with respect to a small number of destinations instead of *any* destination. This scenario typically arises in WSN applications with multiple sinks. RE-CDM allows a sensor node to monitor the connectivity to multiple sink nodes in real time. RE-CDM is a proactive cut detection algorithm. It does not require node localization. Furthermore, nodes need to store only a small amount of data. The contributions of our paper are summarized as follows:

- We extend the notion of the cut detection problem by introducing a *destination-based cut detection* problem.
- We propose a point-to-point cut detection protocol, P2P-CD, that detects a cut between any pair of source and destination. P2P-CD is more energy efficient than RE-CDM when many destination nodes are present, at the price of

higher implementation complexity of the protocol.

- We propose a more light-weight protocol, RE-CDM, that efficiently detects cuts between a source node and a small set of destinations.
- Extensive simulations for large-scale sensor networks demonstrate that our protocols are correct, and efficient when compared with state of art solutions.

The remainder of this paper is structured as follows. In Section II we discuss related work. We formulate the problem and briefly survey existing solution in Section III. The details of our proposed protocols are presented in Section IV, and the performance evaluation results in Section V. We conclude in Section VI with ideas for future work.

II. RELATED WORK

Many researchers have stressed the importance of network partition monitoring problem [12][13][14]. Chong et al. [13] considers the problem as a security issue, mentioning that cuts can be intentionally created in a hostile environment, and nodes must detect them. Cerpa and Estrin [14], in their self-configuring topology scheme, emphasize that the cut detection problem is potentially crucial in many WSN applications, but leave it as future work.

The cut detection problem was first considered in a wired network [4]. Kleinberg et al. [4] introduce the concept of (ϵ, k) -cut, which is defined as a network separation into two sets of nodes, namely $(1 - \epsilon)n$ nodes and ϵn nodes (n refers to the total number of nodes), caused by k independently disabled edges. A set of *agents*, denoted by a set D , is strategically deployed in the network to detect the (ϵ, k) -cut. Each agent exchanges a control packet with other agents periodically. A cut is assumed to be present if the control message loss exceeds some threshold. The authors are interested in the size of D , and prove that the size of the set D is $O(k^3 \frac{1}{\epsilon} \log \frac{1}{\epsilon} + \frac{1}{\epsilon} \log \frac{1}{\delta})$ to detect (ϵ, k) -cut with probability $1 - \delta$. Ritter et al. [6] proposed a cut detection algorithm where a sink node broadcasts an *alive message*. A cut is detected by *border* nodes, which are located on the border of network, if these nodes fail to receive the alive message more than a certain number of times.

Shrivastava et al. [7] has recently introduced a protocol to detect a cut in wireless sensor networks. The work is largely based on [4]. They deploy *sentinels*, a counterpart of *agents* in [4], to detect ϵ -cut, which is defined as a linear cut that separates the network into two parts, where one part has at least ϵ -fraction of total nodes. They aim to minimize the number of sentinels based on the assumption that in sensor networks, linear-shaped or other geometric shaped cuts are more likely to happen, rather than the cut with k independent edge failures. They prove that $O(\frac{1}{\epsilon})$ sentinels are required to detect ϵ -cut with $\epsilon < 1$. The limitations of their cut detection algorithm is that they consider only the linear cuts, being unable to detect arbitrarily shaped cuts. Additionally, the algorithm is a centralized solution, requiring global topology information.

Barooh et al. [9] addresses the problems of previous cut detection algorithms. The Distributed Source Separation

Detection (DSSD) algorithm is fully distributed and detects arbitrarily shaped cuts. A positive scalar value, called *state*, is maintained by each node. The state of each node is updated based on the states of its immediate neighbors. If a node is connected to a sink, its state converges to some positive value. Otherwise, its state rapidly drops to zero. The DSSD algorithm, however, suffers from control message overhead, since the algorithmic iterations for convergence depends on the degree of the network. Won et al. [8] introduced an energy efficient solution that minimizes the iteration count for convergence, thereby minimizing the control message overhead. The main idea is to run the DSSD algorithm on the overlay network consisting of a small number of representative nodes, called *leaders*. The overlay network's degree is at most 4, allowing the optimal convergence rate. However, these algorithms detect cuts to a single sink node. We extend the cut detection algorithm to the cases with multiple sinks, and even further for any destinations.

III. PRELIMINARIES AND PROBLEM FORMULATION

We consider a two dimensional network, represented as a connected graph $G_V = (V, E)$, where $V = \{v_1, v_2, \dots, v_n\}$ is a set of deployed sensor nodes, and E represents the set of links between nodes in V . We denote the set of sink nodes (i.e., base stations) by $S = \{s_1, s_2, \dots, s_n\}$, $S \subseteq V$. We assume that each node knows its location either from an onboard GPS or by employing node localization protocols [15]. We assume that a location-based routing protocol is available, such as GPSR [16]. A set $N_i \subseteq V$ denotes the immediate neighbors of a node $v_i \in V$. $C_v(G)$ represents the connected component of G that contains a vertex v . From here on we will use the terms “source” and “destination” for the sender/receiver pair of a unicast communication. As it will become clear later, destination nodes can be either sink (i.e., base station) nodes, or peer nodes.

Now we are ready to formally define the “Destination-based Cut Detection” problem: Consider a set of destinations, denoted by $T = \{t_1, t_2, \dots, t_n\}$, where $T \subseteq V$. How can a source node $v_i \in V$ determine whether t is in $S_{v_i}(G_V)$, in an energy efficient manner? Informally, we aim to develop energy efficient protocols that allow a node to find its connectivity to each $t \in T$.

Before presenting our solutions to the destination-based cut detection problem, we briefly describe background material. The DSSD algorithm [9] monitors the connectivity of a node to a single sink, say $s_1 \in S$. For ease of presentation, we assume that $v_1 \in V$ is s_1 . Each node $v_i \in V \setminus \{v_1\}$ maintains a positive scalar $v_i(k)$, called the *state*, which is updated at each iteration of the algorithm as the following, where k refers to the iteration counter.

$$v_i(k+1) = \frac{\sum_{v_j \in N_i} v_j(k)}{|N_i| + 1}.$$

The state of a sink node v_1 is updated slightly differently as the following.

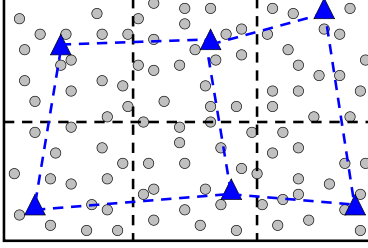


Figure 1. An illustration of the virtual grid network, with leaders, depicted by triangles, elected in each grid cell. The communication among leaders, depicted by dotted lines, is multi-hop.

$$v_1(k+1) = \frac{\sum_{v_1 \in N_1} v_1(k) + \omega}{|N_1| + 1}$$

where ω , called the “sink strength”, is a system parameter. The algorithm proceeds in iterations, and each node updates its state. If there is no cut in the network, the state of a node converges to some positive value, otherwise, the state rapidly converges to 0, allowing a node to detect a cut.

The RE-CD [8] algorithm was proposed for reducing the overhead of DSSD. In RE-CD, as shown in Figure 1, a network is divided into a grid of clusters. In each cluster a leader is elected. In particular, the sink becomes a leader for the cluster that it belongs to. The DSSD algorithm is then executed on the virtual grid network consisting of the leaders (represented as triangles in Figure 1) and the virtual links between them. RE-CD minimizes the convergence rate of the DSSD algorithm, as the number of neighbors for each leader is at most 4 due to the grid network topology (note that the convergence rate of the DSSD algorithm depends on the maximum degree of the network [9]). RE-CD is energy efficient, as only a subset of nodes participate in the cut detection process.

IV. DESTINATION-BASED CUT DETECTION

A. Main Ideas

We propose two cut detection protocols, particularly designed for detecting cuts with respect to a set of destinations. Our first protocol, a point-to-point cut detection (P2P-CD), is designed to solve a more general cut detection problem. Specifically, P2P-CD enables a sensor node to determine the connectivity to any destinations. P2P-CD is based on the knowledge of partial global topology (i.e., P2P-CD requires the information on the shape of the boundary of the cut region, a separated segment of a network caused by a cut). Thus, one issue for this algorithm is to compactly represent the boundary of the cut region. Figure 2 illustrates the general idea on how P2P-CD works. There is a cut in the middle of the network separating the network into two cut regions, denoted by A and B . In P2P-CD, the boundary of cut region is represented as a set of line segments. By connecting the line segments, P2P-CD yields a set of vertices of a polygon covering the cut region. The locations of the vertices of the polygon are distributed to the nodes in the cut region. Based

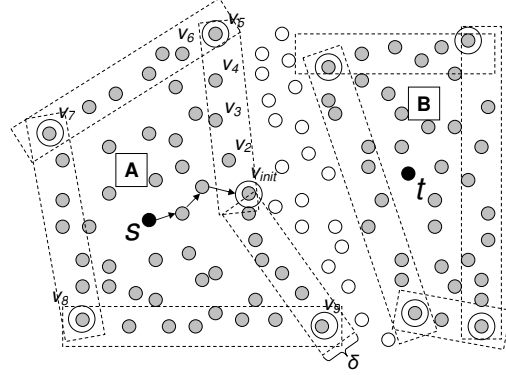


Figure 2. An illustration of the Point-to-Point Cut Detection (P2P-CD) algorithm. A packet initiated by source node S reaches the boundary of the cut, v_{init} in the middle of the figure. During the Cut Boundary Abstraction, the boundary of the disconnected region is detected and abstracted. Information about the abstracted boundary is sent to all nodes within the boundary, which aids them in identifying connection/disconnection from any other node in the network.

on this information, a set of received polygons (there might be multiple cuts in the network), sender s computes whether destination t is reachable or not. The second cut detection protocol we propose, RE-CDM, is suitable for scenarios in which the number of destinations is small (e.g., a set of a few sink nodes). RE-CDM can be used by sensor nodes to autonomously determine connectivity to multiple sink nodes. This protocol is distributed, thus not requiring global topology information. However, it is suitable only for the applications with the small number of potential destinations, as its overhead grows with the number of destinations.

B. Point-to-Point Cut Detection

The point-to-point cut detection (P2P-CD) protocol enables each node in the network to determine the connectivity to any destination. The protocol executes in two main steps. In the first step, the *Cut Boundary Abstraction*, the boundary of the cut region is identified and represented as a polygon $P = \{p_1, p_2, \dots, p_n\}$, where each element of P represents the boundary node corresponding to the vertex of P . As shown in Figure 2, the polygon for cut region A is $P = \{v_{init}, v_5, v_7, v_8, v_9\}$. The polygon P is then broadcast to the nodes in the cut region corresponding to P . In the second step, the *Cut Detection*, any node determines whether a given destination is reachable from it, based on its location, the location of the destination, and the set of polygons $\mathbb{P} = \{P_1, P_2, \dots, P_n\}$ that the node is aware of. Note that a node might receive multiple polygons (e.g., see Figure 3(a) and Figure 3(b)). The following subsections discuss the details of each step of the protocol.

1) *Cut Boundary Abstraction*: The cut boundary abstraction algorithm aims to abstract the boundary information of the cut region. We call the nodes surrounding the boundary of a cut region *boundary nodes*. Our algorithm uses similar technique used in [17] to concisely represent the boundary of the cut region. When a destination is unreachable, a packet would

Algorithm 1 Cut Boundary Abstraction (for v_i)

Input: F_1, F_2, δ , and p_i .

```
1: if  $v_i \neq v_{init}$  then
2:   Cut id  $\leftarrow$  id of initiator.
3:   //  $\square\delta$ : a rectangle with width  $\delta$ .
4:   if  $\forall p \in F_2 \cup \{p_i\}, p$  is in  $\square\delta$  then
5:      $F_2 \leftarrow F_2 \cup \{p_i\}$ .
6:     forward.
7:   else
8:      $F_2 \leftarrow \emptyset$ .
9:      $F_1 \leftarrow F_1 \cup \{\text{the last element in } F_2\}$ .
10:    forward.
11:  end if
12: else
13:    $P \leftarrow F_1$ .
14:   broadcast  $P$ .
15: end if
```

reach one boundary node, say v_{init} . Using the right-hand rule of face routing, this packet travels along the boundary of the cut region until it reaches again v_{init} , thus detecting the existence of a cut [16]. In particular, we call such node v_{init} the *initiator*. The initiator then sends a probing packet that travels around the boundary of the cut region. The probing packet contains two fields. The first field is used to store the locations of the vertices of the polygon representing the boundary of the cut region. We denote the set of such locations by the ordered set F_1 . The second field contains all the locations of visited nodes. We denote the locations of the visited nodes by the ordered set F_2 .

Algorithm 1 depicts the cut boundary abstraction process. Upon receiving the probing packet, a node marks itself as a boundary node and sets the ID of the cut as the node ID of the initiator. The node then finds a rectangle with width δ , a system parameter, that can cover all the locations in the second field, including the location of the current node. If such rectangle exists, the location of the current node is appended to the end of the second field of the probing packet, and the packet is forwarded to the next boundary node (Line 2-6). If such rectangle does not exist, all the locations in the second field are deleted, and the last element in F_2 is appended to the end of the first field (Line 8-9). The current node then forwards the packet to the next boundary node. Note that, in order to keep the size of set F_2 manageable, if the size of F_2 exceeds a threshold, F_2 is emptied and the last element of F_2 is appended to the end of F_1 . Finally, when the probing packet finishes traversing the boundary, we get a set $P = \{p_1, p_2, \dots, p_n\}$ in the first field of the probing packet, representing the polygon covering the cut region. This information is then broadcast to the nodes in the cut region (Line 14), and used by the nodes during the second step of the protocol, namely the Cut Detection.

Consider Figure 2 for an example. In this figure, source s attempts to send a packet to destination t . This packet

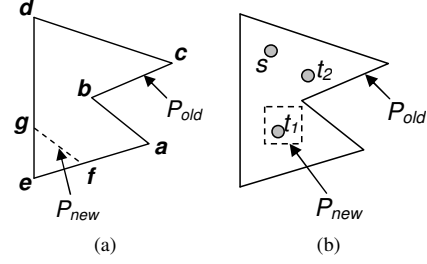


Figure 3. (a) A cut $g-f$ sharing the boundary of a previous cut. (b) An independent cut, inside an existing cut, with a set of source and destination nodes.

then reaches node v_{init} and is routed along the boundary of the cut region A according to the right hand rule of face routing. The packet then returns to node v_{init} starting the cut abstraction process by sending a probing packet to v_2 . When the probing packet reaches v_6 , the first field of the probing packet contains $F_1 = \{v_{init}\}$ and the second field contains $F_2 = \{v_{init}, v_2, v_3, v_4, v_5\}$. The node v_6 then examines if there exists a rectangle with width δ that can hold the locations in $F_2 \cup \{v_6\}$. Since there does not exist such square box, the points in F_2 are deleted and F_1 becomes $\{v_{init}, v_5\}$.

More cuts might occur after existing cuts have been detected. Such cut either shares the boundary of the previously detected cut(s) as shown in Figure 3(a), or is an independent cut as shown in Figure 3(b). In the former case, the probing packet reaches the boundary node of previously discovered cut(s). The cut ID(s) of the previously detected cut(s) is recorded in the probing packet. When the probing packet returns to the initiator, along with the set P , the set of recorded cut IDs, called the UPDATE_INDEX set, is encoded in the broadcast packet that is distributed to the nodes in the cut region. The update bit of the broadcast packet is then set to make the nodes in the cut region know that the previous polygons having the cut IDs specified in the UPDATE_LIST need to be updated. For the latter case, the same boundary abstraction algorithm is used. One difference is that when the probing packet returns to the initiator, the initiator knows that it has already received some boundary information (i.e., $\mathbb{P} \neq \emptyset$). The initiator then sets the addition bit of the broadcast packet, so that the nodes in the cut region add a new polygon in \mathbb{P} .

2) *Cut Detection*: When the cut boundary abstraction process is finished, each node in the cut region recognizes the cut boundary as a polygon, represented as a set $P = \{p_1, \dots, p_n\}$. Given the locations of source s and destination t , and the collection of polygons, \mathbb{P} , the Cut Detection determines whether destination t is reachable from source s . To find the connectivity between any pair of source and destination, we borrow an idea from the point-in-polygon (PIP) problem in the computational geometry that finds whether a point is inside a given polygon or not. There are two well known algorithms to solve this problem: ray casting algorithm and winding number algorithm [18]. We choose to use the ray casting algorithm, because the winding number algorithm involves

Algorithm 2 Cut Detection

Input: p_s, p_t, \mathbb{P} and UPDATE_INDEX

- 1: upon receiving the broadcast packet:
- 2: **if** update type **then**
- 3: $\mathbb{P} \leftarrow \mathbb{P} \setminus \{P_i\}, \forall i \in \text{UPDATE_INDEX}$
- 4: $\mathbb{P} \leftarrow \mathbb{P} \cup P.$
- 5: **else**
- 6: $\mathbb{P} \leftarrow \mathbb{P} \cup P.$
- 7: **end if**
- 8: upon having a packet to destination at p_t :
- 9: **if** $\forall P \in \mathbb{P}, (PIP(P, p_s) \cdot PIP(P, p_t) > 0)$ **then**
- 10: t is reachable.
- 11: **else**
- 12: t is not reachable.
- 13: **end if**

costly operations [18] that are not feasible for the sensor nodes with constrained computational capability.

Ray casting algorithm is as follows. Given a point p , the algorithm finds how many sides of the polygon intersect with the y threshold of the point p . If there are odd times of intersections on each side of p , p is inside the polygon; otherwise, if there are even times of intersections on each side of p , p is outside the polygon. Figure 4 illustrates an example. The point p has 3 intersections on its right side and 3 intersections on its left side. Thus p is determined to be inside the complex polygon. We denote the ray casting algorithm by $PIP(P, p)$, where P refers to a polygon, and p is the point to be tested. We define that if p is inside P , $PIP(P, p) > 0$, otherwise $PIP(P, p) < 0$.

Algorithm 2 depicts the Cut Detection step of the protocol. As we described, the type of the broadcast packet can be either the update type or addition type. If the packet is the update type, we delete the polygons having the cut IDs specified in the UPDATE_INDEX, and add P to \mathbb{P} (Line 2-4). For example, in Figure 3(a), the polygon $\{a, b, c, d, e\}$ is deleted and a polygon $P = \{a, b, c, d, g, f\}$ is added in \mathbb{P} . If the control message is the addition type, it simply adds P to collection \mathbb{P} (Line 6). If there is a packet to send, the Cut Detection tests if the destination is reachable. Specifically, given the location of the source, p_s , and the location of the destination, p_t , the algorithm checks if the following condition holds for each $P \in \mathbb{P}$: $PIP(P, p_s) \cdot PIP(P, p_t) > 0$. If the condition holds for all P , p_s and p_t are connected (Line 9-13).

Consider Figure 3(b) as an example. In this figure, we have two cuts, P_{old} and P_{new} , source s , and two destinations t_1 and t_2 . Since both s and t_1 are inside P_{old} , $PIP(P_{old}, p_s) \cdot PIP(P_{old}, p_{t_1}) > 0$. However, while s is outside P_{new} , t_1 is inside P_{new} , which gives $PIP(P_{new}, p_s) \cdot PIP(P_{new}, p_{t_1}) < 0$; thus t_1 is not reachable from s . On the other hand, for t_2 , $PIP(P_{old}, p_s) \cdot PIP(P_{old}, p_{t_2}) > 0$ and $PIP(P_{new}, p_s) \cdot PIP(P_{new}, p_{t_2}) > 0$, thereby t_2 being reachable from s .

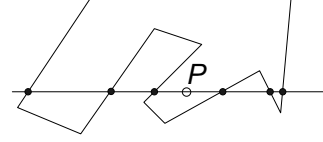


Figure 4. Example of the ray tracing algorithm, with a point p determining if it inside or outside a polygon, by counting the intersection points to the left and right of a line passing through it. In this example, the number of intersections to each side is odd, thus the point is inside the polygon.

C. Energy Efficient Cut Detection for Multiple Sinks

The energy efficient cut detection for multiple sinks, namely RE-CDM, is a more generic solution that builds on our previous work, RE-CD [8]. RE-CDM is based on the virtual grid network consisting of the leaders and the virtual link between them, as in RE-CD. In RE-CDM, however, multiple sink nodes are elected as the leaders, and the leader node now maintains a set of states, as opposed to RE-CD, in which a single state is maintained. Each state value represents the connectivity to a sink node. Note that although multiple sink nodes are typically deployed in such a way that they cover as many sensor nodes as possible, thereby being distant with each other, if we have more than one sink node in the same grid, the one with higher residual energy becomes the leader. The set of states for multiple sinks are updated at each iteration of the algorithm. New states are then encoded in the state message, which is then sent to the neighboring leaders. In essence, RE-CDM overlays the multiple executions of RE-CD for each sink, while using a single state message.

We describe RE-CDM more formally. Consider multiple sink nodes $S = \{s_1, s_2, \dots, s_n\}$. Let a set $L = \{\ell_1, \ell_2, \dots, \ell_n\}$ be the leaders, and N_i^ℓ be the neighboring leaders of a leader ℓ_i (i.e., $|N_i^\ell| \leq 4$). Each leader node ℓ_i maintains a set of states, each of which corresponds to a sink $s \in S$ and is denoted by $\ell_i(s^k)$, where k is the iteration count of the RE-CDM algorithm. At each iteration of the algorithm, each leader node ℓ_i updates the set S as the following.

$$\text{For all } s \in S, \ell_i(s^{k+1}) = \frac{\sum_{\ell_j \in N_i^\ell} \ell_j(s^k)}{|N_i^\ell| + 1}.$$

Each node $\ell_i \in S$ update the set S as the following.

$$\text{For all } s \in S, \ell_i(s^{k+1}) = \frac{\sum_{\ell_j \in N_i^\ell} \ell_j(s^k) + \omega}{|N_i^\ell| + 1}$$

where ω is a system parameter. A state message is now sequentially encoded with the states for multiple sink nodes (i.e., from $\ell_i(s_1^{k+1})$ to $\ell_i(s_n^{k+1})$), and sent to adjacent leaders.

Despite its scalability and energy efficiency, RE-CDM might not work efficiently for large number of sink nodes, as the state message size grows depending on the number of sinks.

V. PERFORMANCE EVALUATION

We evaluate the performance of proposed protocols by simulations. We implement P2P-CD and RE-CDM in C++, mainly

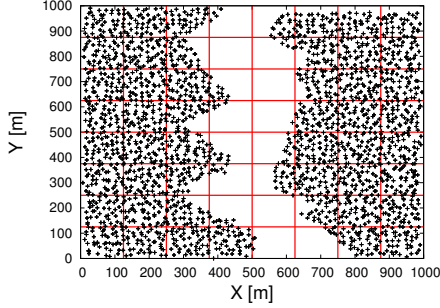


Figure 5. Experimental setup depicting 2,500 nodes deployed in a $1,000 \times 1,000 \text{m}^2$ area and the cut in a network. A superimposed 8×8 grid is used for selecting leader nodes for RE-CDM.

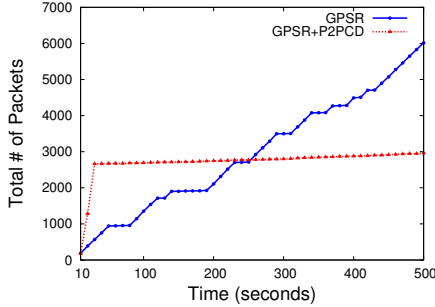


Figure 6. Total number of packets transmitted as a function of time, for GPSR and GPSR enhanced with P2P-CD. As shown, the enhanced version incurs an upfront cost for detecting the cut, cost that is amortized over time, when compared with the cost incurred by GPSR every time a packet is sent towards a non-reachable destination.

focusing on the topological behavior of the protocols. We randomly deploy 2,500 sensor nodes in the $1,000 \times 1,000 \text{m}^2$ network region with a cut, as shown in Figure 5. Communication range is varied from 35m to 75m, resulting in the average number of neighbors from 10.32 to 41.36. We ensure that the network is connected. An event occurs every 10sec at a random position, and then a node nearest to the event reports the event data to a random destination. To simulate the energy consumption, we assume that each node has a radio with 250kbps data rate, and maximum packet size of 128B, similar to the Zigbee compliant CC2420 [19]. We consider the following metrics: the total energy consumption as the number of packet transmissions, network lifetime, false positive rate (a false report of “unreachable destination”), probing packet size and packet loss rate (due to disconnection). We vary the following parameters: δ , communication range, the number of sinks (for RE-CDM) and operation time. We compare the performance of our proposed protocols with GPSR [16] and RE-CD [8]. The details of the experimental results are described in the following subsections.

A. Energy Consumption

P2P-CD protocol incurs communication overhead. Specifically, a probing packet is sent along the boundary of the cut region, and the set P needs to be flooded to the nodes in the cut region. However, this overhead is compensated by

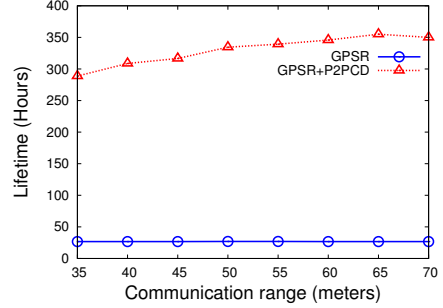


Figure 7. Network lifetime for GPSR and GPSR enhanced P2P-CD. The slight increase in the network lifetime for GPSR enhanced with P2P-CD comes from the smaller number of packet transmissions, due to a longer communication range.

avoiding the unnecessary packet transmissions to unreachable destinations. This experiment is designed to see how much energy P2P-CD can save when the protocol is coupled with GPSR. We measure the accumulated total number of packet transmissions as the operation time elapses. Figure 6 depicts the results. The abrupt increases in the total number of transmissions for GPSR+P2PCD at 10sec and 20sec represent the overhead for identifying the cuts. Due to this overhead, the total number of transmissions of GPSR+P2PCD is higher than GPSR until about 250sec. However, while the total number of transmissions of GPSR+P2PCD gradually increases, GPSR has frequently arising rapid increases in the total number of transmissions, caused by attempting to send a packet to unreachable destination. As a result, after 250sec, GPSR exhibits the higher number of packet transmissions.

B. Network Lifetime

The higher number of packet transmissions in GPSR when compared with GPSR+P2PCD would decrease the network lifetime. We define the network lifetime as the elapsed operation time until one node first dies. We assume that a sensor node is powered by a single AA battery which has capacity of 2000mWh. We measure the network lifetime by varying the communication range. The higher communication range helps to increase the network lifetime by using less number of packet transmissions. As shown in Figure 7, we observe slight increases in network lifetime for both GPSR and GPSR+P2PCD as we increase the communication range. When compared with the network lifetime of GPSR+P2PCD, we observe that the network lifetime of GPSR is much worse than we expected. The main reason is that the number of packet transmissions to unreachable destinations particularly exhaust the energy of the nodes around the boundary of the cut region.

C. False Positive Rate

In our protocol we represent the boundary of the cut region as a polygon. However, a polygon might not be able to accurately represent the boundary of the cut region, causing false positives. To be more specific, the false positive happens when the line connecting two adjacent vertices of a polygon

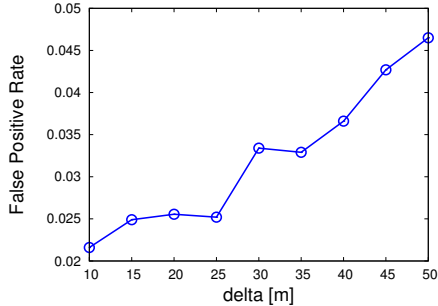


Figure 8. Impact of parameter δ on the rate of false positives. As shown, a larger δ means a less precise approximation for the cut boundary, hence imprecise determination for destination reachability (i.e., some destinations may be deemed not reachable, when they are).

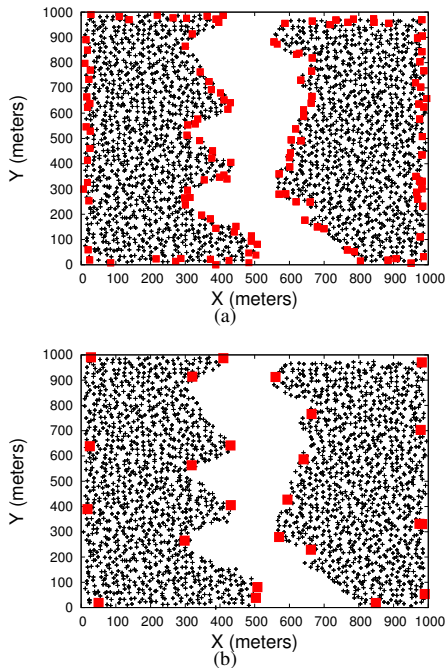


Figure 9. (a) A cut abstracted with $\delta = 10$ meters. As shown a smaller δ allows a more precise approximation of the cut boundary. This is done, however, at the expense of more points describing the cut boundary, an overhead. (b) A cut abstracted with $\delta = 50$ meters. As shown, fewer points/rectangles are used for describing the cut boundary, at the risk of more false positives, as shown in Figure 8.

does not cover all the nodes in the cut region. The parameter, δ , is provided to adjust the accuracy. Smaller δ values allow more precise representation of the boundary, while incurring higher overhead. We measure the number of false positives from 10,000 random source and destination pairs. Figure 8 shows the results. As expected, the false positive rate increases as the δ value increases. The false positive rate is, however, relatively small.

D. Packet Size

As we have seen in Section V-C, if we use a smaller δ value, the false positive rate is decreased. However, smaller δ

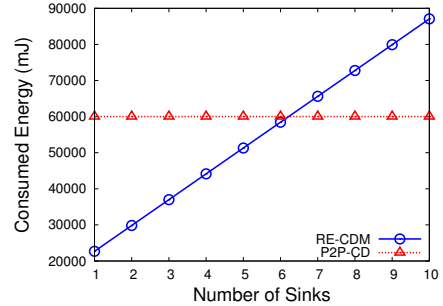


Figure 10. Energy expenditure for cut detection. As shown, if the number of sinks is smaller than 6, RE-CDM is more energy efficient. For a number of sinks larger than 6, the overhead of RE-CDM (a proactive scheme for detecting cuts), becomes too large, and instead, it is more efficient to discover the cuts reactively.

values result in the higher number of vertices of the polygon representing the boundary of the cut region. An increase in the packet size causes higher energy consumption. In this experiment, depicted in Figure 9(a), we measure the packet size as the number of vertices of polygons used to represent the two cut regions. Table I presents the results.

Table I
THE SIZE OF P , IN TERMS OF NUMBER OF VERTICES.

δ	Cut Region(L)	Cut Region(R)
10	69	62
30	16	13
50	12	11

As shown, if we use higher δ values, the packet size becomes smaller. However, higher δ values, at the same time, increase the false positive rates. Figure 9(a) and 9(b) show how the vertices for the polygon are chosen for different δ values.

E. RE-CDM and P2P-CD Energy Consumption

In order to test the performance of our RE-CDM protocol, we divide the network into 8×8 grid cells, each of which has the size of $125 \times 125 \text{m}^2$. Different number of sinks are randomly deployed throughout the network. We measure consumed energy for RE-DCM and P2P-CD until the cut is detected by varying the number of sinks. We assume that P2P-CD detects a cut when the first event happens, and for RE-CDM, we assume that a node wakes up when it needs to send the state message and sleeps between state updates. Figure 10 depicts the results.

We observe that RE-CDM's energy consumption linearly increases as we increase the number of sinks. One reason is that the packet size becomes larger as there are more sink nodes for which cuts should be detected. As expected, for the smaller number of sinks, RE-CDM is more energy efficient, but at some point P2P-CD starts to outperform RE-CDM. These results illustrate the tradeoff, i.e., energy consumption vs. protocol implementation complexity, faced when using one of the two protocols. RE-CDM is a simpler protocol, but might consume more energy than P2P-CD, a more sophisticated protocol, in some scenarios.

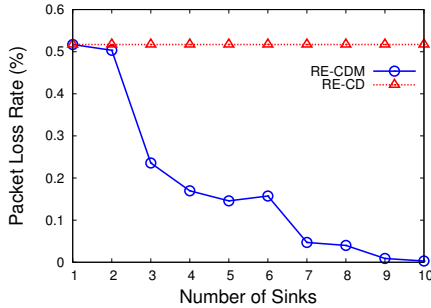


Figure 11. Packet loss rate as a function of number of sinks. If there are no sink nodes in the disconnected region, then the packet intended for a sink node is lost. As shown, for larger number of sink nodes, RE-CDM is able to reduce the packet loss to almost 0.

F. RE-CDM: Packet Loss Rate

When an event occurs, a sensor node near the event reports data to the closest sink node. When there is no sink node in the cut region the packet is lost. For each set of experiment, we redeploy the sink nodes randomly. Figure 11 depicts the results. Regardless of the number of sink nodes, RE-CD's packet loss rate is about 50%. The reason is simply the probability that a sink is in one cut region and not in another. In contrast, the packet loss rate for RE-CDM rapidly drops as we have more sink nodes. When there are 10 sink nodes, the packet loss rate becomes 0%. Note that the packet loss rates for 2 to 9 sink nodes are not 0%, because the sink nodes are redeployed for each experiment, so that there are chances of having all sink nodes in the same cut region for smaller number of sink nodes.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we defined and proposed solutions for the destination-based cut detection problem, in which cuts are detected based on a given set of destinations. We first introduce a cut detection protocol for multiple sinks (RE-CDM), useful for WSN applications with multiple sinks. We then extend the number of target destinations and propose the point-to-point cut detection protocol (P2P-CD). In P2P-CD the source node is able to determine if any destination node is reachable or not. As future work, we plan to develop a point-to-point cut detection protocol that does not rely on nodes locations. This will enable us to employ other types of routing protocols than location based.

REFERENCES

[1] S. M. George, W. Zhou, H. Chenji, M. Won, Y. Lee, A. Pazarloglou, R. Stoleru, and P. Barooah, "DistressNet: a wireless Ad-Hoc and sensor network architecture for situation management in disaster response," *IEEE Communications Magazine*, vol. 48, no. 3, Mar. 2010.

[2] T. He, S. Krishnamurthy, L. Luo, T. Yan, L. Gu, R. Stoleru, G. Zhou, Q. Cao, P. Vicaire, J. A. Stankovic, T. F. Abdelzaher, J. Hui, and B. Krogh, "VigilNet: an integrated sensor network system for energy-efficient surveillance," *ACM Transactions on Sensor Networking*, vol. 2, no. 1, pp. 1–38, 2006.

[3] A. Wood, J. Stankovic, G. Virone, L. Selavo, Z. He, Q. Cao, T. Doan, Y. Wu, L. Fang, and R. Stoleru, "Context-aware wireless sensor networks for assisted living and residential monitoring," *Network, IEEE*, vol. 22, no. 4, pp. 26–33, 2008.

[4] J. Kleinberg, "Detecting a network failure," *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, p. 231, 2000.

[5] J. Kleinberg, M. Sandler, and A. Slivkins, "Network failure detection and graph connectivity," in *Proc. of ACM SODA*, 2004.

[6] H. Ritter, R. Winter, and J. Schiller, "A partition detection system for mobile ad-hoc networks," in *Proc. of IEEE SECON*, 2004.

[7] N. Shrivastava, S. Suri, and C. Tóth, "Detecting cuts in sensor networks," *ACM Transactions on Sensor Networks*, vol. 4, no. 2, pp. 1–25, 2008.

[8] M. Won, M. George, and R. Stoleru, "Towards robustness and energy efficiency of cut detection in wireless sensor networks," *Elsevier Ad Hoc Networks*, vol. 9, no. 3, pp. 249–264, 2011.

[9] P. Barooah, "Distributed cut detection in sensor networks," in *Proc. of IEEE CDC*, 2008.

[10] E. Oyman and C. Ersoy, "Multiple sink network design problem in large scale wireless sensor networks," in *Proc. of IEEE ICC*, 2004.

[11] A. Das and D. Dutta, "Data acquisition in multiple-sink sensor networks," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 9, pp. 82–85, July 2005.

[12] V. Park and M. Corson, "A highly adaptive distributed routing algorithm for mobile wireless networks," in *Proc. of IEEE INFOCOM*, 1997.

[13] C.-Y. Chong and S. Kumar, "Sensor networks: evolution, opportunities, and challenges," *Proceedings of the IEEE*, vol. 91, no. 8, pp. 1247 – 1256, Aug. 2003.

[14] A. Cerpa and D. Estrin, "ASCENT: Adaptive self-configuring sensor networks topologies," *IEEE Transactions on Mobile Computing*, vol. 3, no. 3, pp. 272–285, 2004.

[15] R. Stoleru, J. Stankovic, and S. Son, "On composability of localization protocols for wireless sensor networks," *Network, IEEE*, vol. 22, no. 4, pp. 21 –25, july-aug 2008.

[16] B. Karp and H. T. Kung, "GPSR: Greedy perimeter stateless routing for wireless networks," in *Proc. of ACM MOBICOM*, 2000, pp. 243–254.

[17] G. Tan, M. Bertier, and A.-M. Kermarrec, "Visibility-graph-based shortest-path geographic routing in sensor networks," in *Proc. of IEEE INFOCOM*, 2009.

[18] I. E. Sutherland, R. F. Sproull, and R. A. Schumacker, "A characterization of ten hidden-surface algorithms," *ACM Comput. Surv.*, vol. 6, pp. 1–55, March 1974.

[19] *CC2420 Data Sheet*, <http://www.chipcon.com>.