A Destination-based Approach for Cut Detection in Wireless Sensor Networks

Myounggyu Won, *Student Member, IEEE,* and Radu Stoleru, *Member, IEEE*

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 lightweight proactive cut detection algorithm specifically designed for a network scenario with a small set of target destinations. We prove the effectiveness of the proposed algorithms through extensive simulations; specifically, in our network configurations, proposed cut detection algorithms achieve more than an order of magnitude improvement in energy consumption, when coupled with an underlying routing protocol.

Index Terms—wireless sensor networks, topology control, network cut detection

1 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 to 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][10]. 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 [11][12]. In these applications, detecting a cut with respect to one sink node does not necessarily mean that nodes in the disconnected network segment should refrain from reporting data, because they may send the data to other connected sink nodes.

In this article, 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 overhead. A source node, using the distributed information, locally determines whether any given destination is reachable or not.

P2P-CD is a reactive algorithm; in other words, a

• *M. Won and R. Stoleru are with the Department of Computer Science and Engineering, Texas A&M University, College Station, TX 77840. E-mail: mgwon@cse.tamu.edu, stoleru@cse.tamu.edu*

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, the energy efficiency comes at the cost of overhead: 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 also propose a lightweight 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 each sensor node to monitor the connectivity to multiple sink nodes in real time. RE-CDM is a proactive cut detection algorithm, being less energy efficient than P2P-CD. However, it does not require nodes to be localized, and nodes do not need to store data on the partial global topology, which makes a good fit with resource constrained nodes in WSNs. The contributions of our article are summarized as follows:

- We extend the notion of the cut detection by introducing a novel problem called *destination-based cut detection*
- 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 lightweight 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.

The remainder of this article is structured as follows. In Section 2 we discuss related work, and the taxonomy for cut detection schemes. We formulate the problem and briefly survey existing solutions in Section 3. The details of our proposed protocols are presented in Section 4, and the performance evaluation results in Section 5. We conclude in Section 6.

2 Related Work

2.1 Cut Detection

Many researchers stressed the importance of network partition monitoring problem [13][14][15]. Chong et al. [14] considered 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 [15], in their self-configuring topology scheme, emphasized that the cut detection problem is potentially crucial in many WSN applications, but left it as future work.

The cut detection problem was first considered in a wired network [4]. Kleinberg et al. [4] introduced 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] recently introduced a protocol to detect a cut in wireless sensor networks. Their work is largely based on [4]. The protocol deploys *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. The paper aims 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, their algorithm is a centralized solution, requiring global topology information.

Barooah et al. [9][10] addressed the issues that previous cut detection algorithms have. 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 converges to zero. The DSSD algorithm, however, suffers from control message overhead, since the algorithmic iterations for the convergence depends on the degree of the network. Won et al. [8][16] introduced an energy efficient solution that minimizes the iteration count for the 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 degree of the overlay network is at most 4, allowing the minimal convergence rate. However, these algorithms detect cuts with respect to only a single sink node.

In [17], we considered a novel cut detection problem, called the destination-based cut detection, where

Cut detection scheme	criterion 1	criterion 2	criterion 3	properties
Flooding	k_1	1	proactive	k_1 is the number of nodes in the disconnected network segment.
Linear Cut Detection [7]	k_2	1	proactive	centralized; detects only a linear cut. k_2 is the number of sentinels.
DSSD [9]	N	1	proactive	distributed; detects a cut of any shape.
RE-CD [8]	N	1	proactive	cluster-based; more energy efficient than DSSD.
RE-CDM	N	k_3	proactive	extends RE-CD; supports cut detection with respect to k_3 sinks.
P2P-CD	N	N	reactive	detects a cut between any pair of nodes; requires node location

TABLE 1
Taxonomy for cut detection schemes. Existing cut detection schemes are categorized based on three criteria: (a) who detects a cut; (b) detects a cut with respect to which nodes; (c) proactive/reactive.

cuts are detected with respect to any target destination. We then presented two algorithms for this problem: one algorithm is suitable for the large scale network with many peer-to-peer communications; and the other algorithm is a good fit for the network scenario with a small number of target destinations such as the sensor network applications with multiple sink nodes, where sensor nodes send data only to the sink nodes. This article extends our previous work. First, we present a complete taxonomy for existing cut detection schemes not only to provide guidelines for the future research on this topic, but also to clarify the contributions of this article. Second, we consider several practical issues. We propose an algorithm to eliminate false positives caused by errors in describing the geometric structure of a cut in a network. We then take the asynchronous duty cycling into consideration in order to implement more realistic network settings. Under this network setting, we measure more realistic energy consumption. Third, we develop several new metrics for our simulations to show the efficacy of our proposed algorithms.

2.2 Taxonomy for Cut Detection

Systematically organizing the previously introduced cut detection algorithms not only helps better understand the contributions of this article, but also provides guidelines for future research on this topic. We categorize the algorithms based on three criteria. First, we consider which nodes detect a cut. Some algorithms allow only a small subset of nodes to detect a cut, whereas some algorithms allow all nodes in the network to detect a cut. Second, we consider that, with respect to which nodes, a cut is detected. Such "target" nodes might be the sink node, or for some algorithms, all nodes. The last criterion describes whether the cut detection algorithm is proactive or reactive. The proactive solution periodically checks the existence of a cut, whereas the reactive solution runs the algorithm only when there is a cut; thus, a reactive solution is a more energy efficient scheme.

Table 1 summarizes the taxonomy for existing cut detection algorithms. As shown, the most basic type of cut detection algorithms is the flooding. In this scheme, a sink node periodically broadcasts a probing packet throughout the network so that the receivers can check the network connectivity to the sink. One drawback of this scheme is that the nodes in the connected network segment cannot detect a cut; only the k_1 number of nodes in the disconnected network segment can detect a cut. The Linear Cut Detection algorithm proposed by Shrivastava et. al [7] significantly reduces the message overhead caused by broadcasting the probing packet throughout the network, by allowing only a small subset of nodes, called the sentinels, participate in the cut detection process; thus, in this scheme, the k_2 sentinels detect a cut with respect to a sink node. The DSSD algorithm [9] operates in a distributed manner and allows all the N nodes in the network to detect a cut with respect to a sink node. RE-CD [8] algorithm improves the energy efficiency of the DSSD by minimizing the convergence rate of the DSSD algorithm. Despite its better energy efficiency, this algorithm still permits all the nodes in the network detect a cut with respect to a sink node.

At this point, we note that existing algorithms focus on detecting a cut with respect to a single node, the sink node, and furthermore they are all proactive solutions. Two proposed algorithms in this article extend the notion of existing cut detection; specifically, we extend the number of "target" nodes. First of all, RE-CDM is designed to enable nodes to detect a cut with respect to k_3 sink nodes. This algorithm is distributed, but a proactive solution. Our reactive cut detection algorithm, P2P-CD then further extends the second criterion, the number of target destinations, to all the N nodes in a network at the cost of several requirements: 1) each node has to be localized; 2) additional storage space is required; and 3) part of global topology information must be known to the nodes in the network. Despite these requirements, this algorithm, is reactive, thereby being energy efficient.

In sum, P2P-CD realizes the concept of peer-to-peer

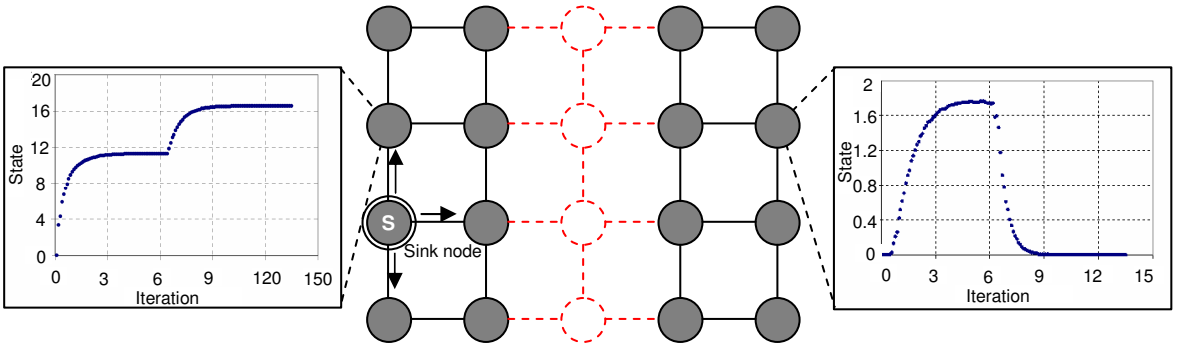

Fig. 1. An illustration of DSSD algorithm. When a node is connected to the sink, its state converges to some positive value. The four nodes in the middle marked as red dotted circles die at iteration 6, creating a cut. After the cut occurs, the state of the node connected to the sink converges to new convergence value (i.e., from approximately 11 to 17); the state of the node that is disconnected from the sink converges to 0, being able to detect the cut.

cut detection. In other words, in P2P-CD, a source node can detect a cut with respect to any destination; the RE-CDM is a more lightweight solution that does not rely on the space and implementation overhead, which suits well for the applications that have a small number of target nodes, such as the applications for WSN with multiple sinks.

3 Preliminaries and Problem Formulation

We consider a two dimensional network, represented as a connected graph $G_V = (V, E)$, where $V = \{v_1, v_2, \cdots, v_n\}$ is a set of deployed sensor nodes, and E represents a set of links between nodes in V. We denote a set of sink nodes (i.e., base stations) by $S = \{s_1, s_2, \cdots, s_n\}$, $S \subseteq V$. We assume that each node knows its location either from an onboard GPS, or by employing node localization protocols [18]. We also assume that a location-based routing protocol is available, such as GPSR [19]. A set $N_i \subseteq V$ denotes the immediate neighbors of a node $v_i \in V$. The term $C_v(G)$ represents the connected component of graph 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 nodes (i.e., base station), or peer nodes.

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

Before presenting our solutions for the destination-based cut detection problem, we briefly describe some background materials. The DSSD algorithm [9] monitors the connectivity of a node to a single sink, say

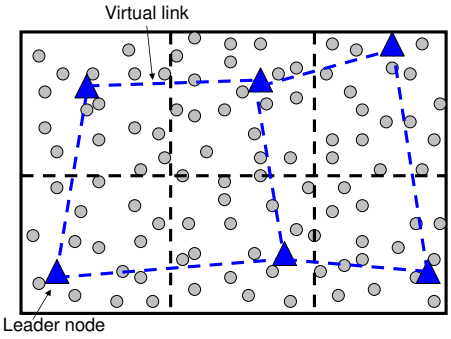

Fig. 2. An illustration of the virtual grid network, with leaders, depicted by triangles, elected in each grid cell. The communication among leaders, depicted by dotted lines, is multi-hop.

$s_1 \in S$. For ease of presentation, we assume that $v_1 \in V$ is s_1. Each node $v_i \in V$ 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.

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

Here the variable 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. Figure 1 illustrates how DSSD algorithm works.

The RE-CD [8] algorithm was proposed for reducing the overhead of DSSD. In RE-CD, as shown in

Figure 2, 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 2) and the virtual links between them. RE-CD minimizes the convergence rate of the DSSD algorithm, because 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, because only a subset of nodes participate in the cut detection process.

4 Destination-based Cut Detection

This section discusses the details of the two cut detection algorithms that we propose.

4.1 Main Ideas

Before presenting the details, we first overview the two proposed algorithms with the tradeoffs between them. Both algorithms are designed for detecting cuts with respect to a given set of destinations. Our first protocol, Point-to-Point Cut Detection (P2P-CD), is designed to provide a solution that enables each node to determine connectivity to any destination. Note that a cut partitions a network into multiple network segments; we call such network segment a *cut region*. P2P-CD is based on the knowledge of partial global topology: it uses the shape and location of a cut region. An important issue for this algorithm is thus to compactly, yet precisely, represent the boundary of a cut region. Figure 3 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 a 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 on the set of received polygons (there might be multiple cuts in the network), nodes determine whether a given destination is reachable or not. The second cut detection protocol we propose, RE-CDM, is suitable for scenarios in which the number of target 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 does not require global topology information, nor the location information, thereby reducing the space and implementation overhead. However, it is suitable only for the applications with the small number of target destinations, because its overhead grows with the increasing number of destinations.

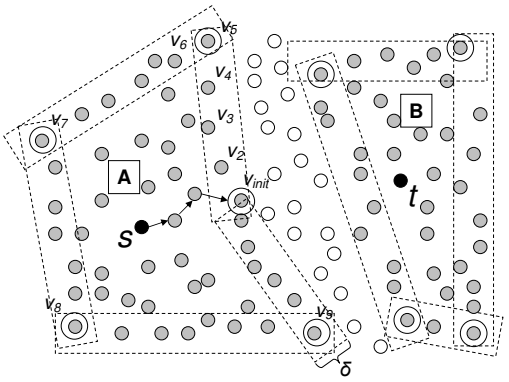

Fig. 3. An illustration of the Point-to-Point Cut Detection (P2P-CD) algorithm. A packet initiated by source node S reaches the boundary of the cut, v_{init} in the middle of the figure. During the Cut Boundary Abstraction, the boundary of the cut region is detected and abstracted. Information about 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.

4.2 Point-to-Point Cut Detection

The point-to-point cut detection (P2P-CD) protocol enables each node in a network to determine the connectivity to any destination. This protocol executes in two main steps. In the first step, the *Cut Boundary Abstraction*, the boundary of a cut region is identified and represented as a polygon $P = \{p_1, p_2, \cdots, p_n\}$, where each element of P is the location of a node that represents the vertex of P. Consider Figure 3 for an example. In this figure, the polygon corresponding to the cut region A is $P = \{v_5, v_7, v_8, v_9\}$. After the polygon P is identified, it is broadcast to the nodes in the cut region corresponding to the polygon P. In the second step, the *Cut Detection* phase, nodes determine whether a destination is reachable based on the following available information: its location, the location of the destination, and a set of polygons $\mathbb{P} = \{P_1, P_2, \cdots, P_n\}$ that the node has received. Note that a node might receive multiple polygons when it is involved with multiple cuts (e.g., as we will later see in Figure 5(a) and Figure 5(b)). Following subsections discuss the details of each step of the protocol.

4.2.1 Cut Boundary Abstraction

The cut boundary abstraction algorithm aims to abstract the boundary information of a cut region. We call the nodes surrounding the boundary of a cut region *boundary nodes*. Our algorithm uses similar technique used in [20] to concisely represent the boundary of a cut region. When a destination is unreachable, a packet would reach one boundary node, say v_{init}. Using the right-hand rule of the face routing, this packet travels along the boundary of the cut region until it reaches again v_{init}, thus detecting the existence

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 δ.
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: $F_1 \leftarrow F_1 \cup \{$the last element in $F_2\}$.
10: forward.
11: **end if**
12: **else**
13: $P \leftarrow F_1$.
14: broadcast P.
15: **end if**

of a cut [19]. 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 a cut region. We denote the set of such locations by an ordered set F_1. The second field contains all the locations of visited nodes. We denote the locations of the visited nodes by an ordered set F_2.

Algorithm 1 depicts the cut boundary abstraction phase. Upon receiving the probing packet, a node marks the ID of the currently detected 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 a 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 a 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. Note that the threshold is determined based on the maximum packet size in order to make the set fit in a packet. Finally, when the probing packet finishes traversing the boundary, we get a set $P = \{p_1, p_2, \cdots, p_n\}$ 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 3 for an example. In this figure, source s attempts to send a packet to destination t. This packet 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 node v_2. When the probing packet reaches node v_6, the first field of the probing packet contains set $F_1 = \{v_{init}\}$ and the second field contains set $F_2 = \{v_{init}, v_2, v_3, v_4, v_5\}$. The node v_6 then examines if there exists a rectangle with width δ that can hold all the locations in set $F_2 \cup \{v_6\}$. Since there does not exist such a rectangle box, the points in F_2 are deleted and F_1 becomes $\{v_{init}, v_5\}$.

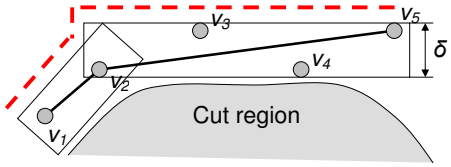

Fig. 4. An illustration of the false positive. According to the P2P-CD algorithm, the line connecting node v_2 and v_5 becomes one of the edges of a polygon covering the given cut region. However, if this edge is used, node v_3 is not covered by the polygon, causing the false positive. The FPE (False Positive Elimination) algorithm finds a new polygon, the two edges of which are represented as dotted lines, that makes the false positive rate be 0%.

As discussed, the boundary of a cut region is compactly represented as a polygon. Despite its concise representation, a polygon, however, might fail to precisely describe the boundary of a cut region, causing false positives. Specifically, a false positive occurs when a polygon fails to cover all the nodes in a cut region. Figure 4 illustrates a scenario where the false positive occurs. This figure shows a fraction of the boundary nodes for cut region A. According to the cut abstraction phase of the P2P-CD algorithm, nodes v_1, v_2, and v_5 serve as the vertices of the polygon representing the boundary of the given cut region. We note that this polygon, however, fails to cover node v_3 that is outside the polygon; thus, when a node attempts to send a packet to node v_3, it will determine that node v_3 is unreachable, although the node is reachable, causing the false positive.

In order to eliminate the false positive, our P2P-CD algorithm is slightly modified at the cost of extra data storage for saving additional vertices. The proposed idea, named the FPE (False Positive Elimination), is simple, yet effective. Instead of constructing a polygon by connecting the two nodes at each end of a bounding box, we build a polygon by connecting the edges of bounding boxes that face the outside of a cut region. This can be easily done by saving the locations of the previous bounding box, and calculating the intersecting points with the currently in-

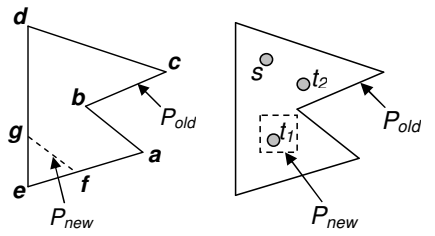

Fig. 5. Illustrations of multiple cuts. (a) A new cut $a-b-c-d-g-f$ shares its boundary with the previously detected cut with the boundary $a-b-c-d-e$ (b) An independent cut represented as the rectangle with dotted lines, inside an existing cut, P_{old}.

vestigated bounding box in the cut abstraction phase. See Figure 4 for an example. In this example, we use a series of dotted lines as the edges of the polygon covering the cut region, rather than using the two line segments, which are $\overline{v_1v_2}$ and $\overline{v_2v_5}$. In Section 5.3, we shall investigate the impact of the false positives and show that this false positive is indeed eliminated when we adopt the FPE mechanism.

4.2.2 Multiple Cuts

Additional cuts might appear after existing cuts have been detected. Such a cut either shares the boundary of previously detected cut(s), or is an independent cut that does not share its boundary with previously detected cuts. Figure 5(a) depicts the former case. The old cut P_{old} represented by the polygon $a-b-c-d-e$ shares its boundary with the new cut $a-b-c-d-g-f$. In this case, when the probing packet reaches the boundary nodes of previously discovered cut(s), the cut ID(s) of previously detected cut(s) is recorded in the probing packet, if it has not been done so. When the probing packet returns to the initiator, the set of recorded cut IDs, called the UPDATE_INDEX set, and the set P representing the polygon for the new cut are encoded in the broadcast packet, which then is distributed to the nodes in the cut region. Upon receiving this broadcast packet with the UPDATE_INDEX set, nodes update their database for the detected cuts. The details of this update process is described in the following subsection.

Figure 5(b) illustrates the latter case where a new cut does not share its boundary with previously detected cuts. In this figure, the rectangle with dotted lines represents the new cut. In this case, the same boundary abstraction algorithm is used to describe the new cut as a polygon. One difference is that when a probing packet returns to the initiator, the initiator sets the addition bit of the broadcast packet, so that the nodes in the cut region add a new polygon to its database, $\mathbb{P}$.

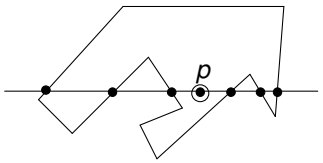

Fig. 6. Example of the ray tracing algorithm, with a point p determining if it inside or outside a polygon, by counting the intersection points to the left and right of a line passing through it. In this example, the number of intersections to each side is odd, thus the point is inside the polygon.

4.2.3 Cut Detection

When the cut boundary abstraction phase is finished, each node in a cut region recognizes the cut boundary as a polygon, represented as a set $P = \{p_1, \cdots, p_n\}$. Given the locations of source s and destination t, and the collection of polygons, $\mathbb{P}$, the Cut Detection phase 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 [21]. We choose to use the ray casting algorithm, because the winding number algorithm involves costly operations [21] that are not feasible for the sensor motes with constrained computational capability.

Ray casting algorithm works 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 6 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 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 phase of the P2P-CD 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, from $\mathbb{P}$, 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 5(a), the polygon $\{a, b, c, d, e\}$ is deleted and a polygon $P = \{a, b, c, d, g, f\}$ is added to $\mathbb{P}$. 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 phase tests if the destination is reachable. Specifically, given the location of the source, p_s, and the location of the destination, p_t,

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: $\mathbb{P} \leftarrow \mathbb{P} \setminus \{P_i\}$, $\forall i \in$ 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**

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 5(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.

4.3 Energy Efficient Cut Detection for Multiple Sinks

As we described, P2P-CD is a suitable protocol for a network with frequent peer to peer communications. However, this protocol maybe an overkill for a network scenario where there are a small number of target destinations, because this protocol requires each node to save the information on partial global topology. For example, in many sensor network applications, sensor nodes transmit the data with perceived phenomenon only to a sink node, or multiple sink nodes. In other words, the number of target destinations is limited. For these applications, we develop a lightweight cut detection algorithm, which relies on neither location information, nor the knowledge on global topology.

The energy efficient cut detection for multiple sinks, called the 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 leaders, and each leader node now maintains a set of states, as opposed to RE-CD, in which a single state is maintained. Each state value

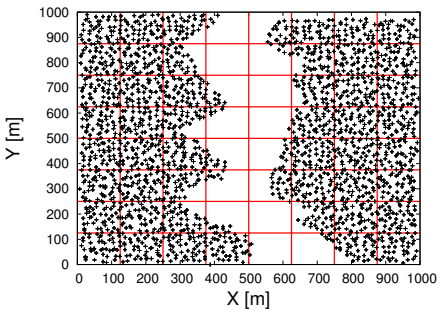

Fig. 7. Experimental setup depicting 2,500 nodes deployed in a 1,000×1,000m^2 area and the cut in a network. A superimposed 8×8 grid is used for selecting leader nodes for RE-CDM.

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 cell, the one with higher residual energy becomes the leader. The set of states for multiple sinks are updated at each iteration of the algorithm. The updated 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 $S = \{s_1, s_2, \cdots, s_n\}$. Let a set $L = \{\ell_1, \ell_2, \cdots, \ell_n\}$ 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 $\ell_i \notin S$ 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.

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

Here, 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, because the state message size grows, and may be split into multiple packets depending on the number of sinks.

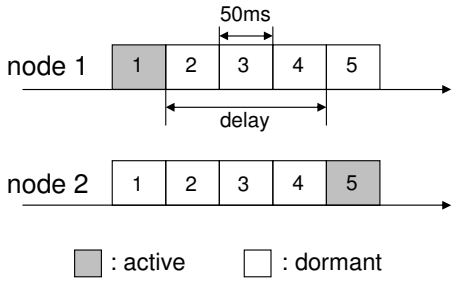

Fig. 8. Illustration of the source of energy consumption. When node 1 has a packet for node 2, node 1 waits until node 2 becomes active at slot 5. Node 1 then turns on its radio at slot 5 and transmits a packet to node 2; thus, this packet transmission process requires one additional active slot (slot 5) besides scheduled ones.

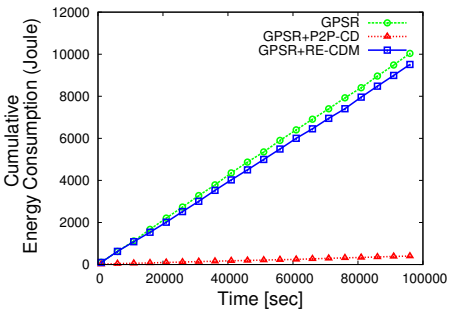

Fig. 9. Accumulated energy measured for long time. The gap of energy consumption between GPSR and GPSR+P2P-CD comes from the packet transmissions for unreachable destinations.

5 Large-scale Simulations

We evaluate the performance of the proposed protocols through simulations. We implement P2P-CD and RE-CDM in C++, mainly focusing on the topological behavior of the protocols. We randomly deploy 2,500 sensor nodes in a network of $1{,}000 \times 1{,}000 m^2$ region with a cut, as shown in Figure 7. The communication radius of a node is varied from 35m to 75m, resulting in an 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 location, and then a node nearest to the event reports the event data to a random destination.

For accurate energy consumption estimation, we consider an asynchronous duty cycle network, where each node has a randomly generated periodic schedule. Nodes periodically wake up and sleep based on the schedule. We assume that a node knows about the schedules of its neighbors, or the parameters used for the pseudo-random schedule generator like [22] (i.e., these parameters are used to deduce the schedules of its neighbors). In order to coordinate the schedule of a node with its neighbors, we assume that a local synchronization is implemented. The local time synchronization can be implemented by using a MAC-layer time stamping technique [23]. The MAC-layer time stamping technique achieves the accuracy of 2.24μs at the cost of exchanging a few bytes of packet transmissions with its immediate neighbors every 5 minutes. This accuracy is by far enough for our asynchronous duty cycle scheme. Each period of the schedule consists of 100 time slots. The unit time for each slot is 50ms. The number of time slots during which a node is active varies according to the duty cycle ratio, which is a system parameter.

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 [24]. We consider the following metrics: total energy consumption, network lifetime, false positive rate (a false report of "unreachable destination"), probing packet size, detection delay, and control packet overhead. We vary the following parameters: δ, communication radius, and duty cycle ratio. We use GPSR as the underlying routing protocol, and build different cut detection algorithms on top of it. We compare GPSR+P2P-CD, GPSR+RE-CDM, GPSR+Flooding, and GPSR with no cut detection scheme. The details of experimental results are described in the following subsections.

5.1 Energy Consumption

In an asynchronous duty cycle network, nodes consume energy when they are awake for either idle listening, or data transmission. Regardless of the protocols run on the network, the same amount of energy is spent for idle listening, because the predefined working schedules determine the energy consumption for idle listening; thus, we consider only the data transmission for the following reasons. First, the current draw for receive mode and transmit mode are different (the difference depends on a radio module; for some radio modules, more current is draw for receive mode). More important reason is that when a node transmits a packet, it may have to use an additional slot to wake up its radio. Figure 8 illustrates this concept. When node 1 attempts to transmit a packet to its neighbor, node 2, it has to turn on its radio at slot 5, which was originally scheduled for sleep. This causes extra energy consumption.

This experiment is designed to see how much energy P2P-CD can save when P2P-CD is coupled with GPSR. The communication radius was set to 70m, δ to 60m, and duty cycle ratio to 1%. We measure accumulated energy consumption in Joules for GPSR, GPSR+P2P-CD, and GPSR+RE-CDM, as a function of operation time. Figure 9 depicts the results. P2P-CD significantly reduces the energy consumption by preventing packet transmissions to unreachable destinations. RE-CDM is not much helpful for energy saving in this simulation scenario, because it detects a cut with respect to a small set of destinations; in

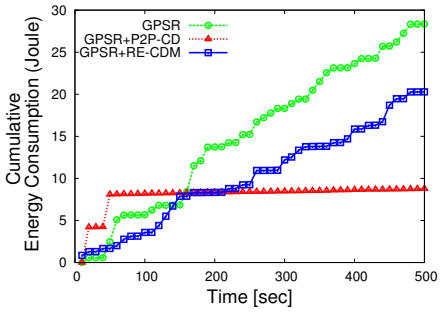

Fig. 10. Communication overhead for P2P-CD to detect a cut. As shown P2P-CD incurs an upfront cost for detecting the cut, cost that is amortized over time, when compared with the cost incurred by GPSR every time a packet is sent towards a non-reachable destination.

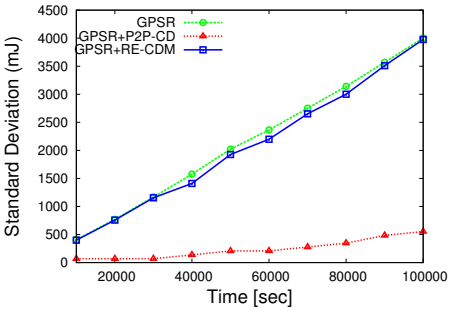

Fig. 11. Standard deviation of energy consumption of the nodes in the network. The standard deviations of GPSR and GPSR+RE-CDM are high compared with GPSR+P2P-CD, because packets destined to unreachable destinations always travel along the boundary nodes, exhausting the energy of the boundary nodes.

our network setting of 8 by 8 grids, it detects a cut with respect to 64 nodes. Moreover, the periodic state message exchanges make RE-CDM less energy efficient than P2P-CD. However, as we prove later, for network scenarios with a small set of destinations, such as a WSN application with multiple sinks, RE-CDM is as energy efficient as P2P-CD.

The P2P-CD protocol incurs communication overhead. Specifically, a probing packet is sent along the boundary of a cut region, and a message containing the obtained set P needs to be flooded to the nodes in the cut region. Figure 10 shows how this control packet overhead affects the energy efficiency. The abrupt increases in the energy consumption for GPSR+P2PCD at 10sec and 40sec represent the overhead for identifying the cuts. Due to this overhead, consumed energy for GPSR+P2PCD is higher than GPSR until about 150sec. However, this overhead is compensated by avoiding unnecessary packet transmissions to unreachable destinations; while consumed energy of GPSR+P2PCD gradually increases, GPSR has frequently arising rapid increases in the energy consumption, caused by attempting to send a packet to unreachable destinations. As a result, after 150sec, GPSR exhibits higher energy consumption.

5.2 Network Lifetime

Another important issue (in addition to high energy consumption) caused by a cut in a network, is unbalanced energy consumption. A packet destined to an unreachable destination always travels around the boundary of a cut region, thereby exhausting the energy of the nodes on the boundary faster than other nodes. In order to verify this unbalanced energy consumption, we measure the standard deviation of energy consumption of all the nodes in the network. Figure 11 depicts the results. As expected, the unbalance in energy consumption of GPSR is much worse than GPSR+P2P-CD, and slightly worse

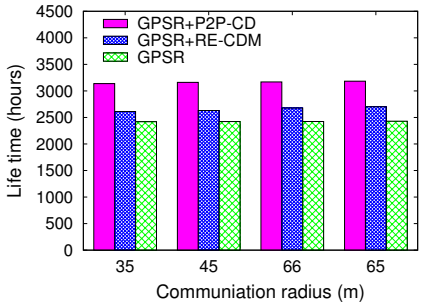

Fig. 12. Network lifetime for GPSR, GPSR+RE-CDM, and GPSR+P2P-CD. The cut detection schemes elongate the network lifetime when coupled with an underlying routing protocol.

than GPSR+RE-CDM. This unbalance deteriorates as operation time elapses.

This unbalanced energy consumption directly affects the network lifetime, which is defined as the elapsed time until a node first dies. In this set of experiments, we are interested in obtaining the expected network lifetimes for GPSR, GPSR+RE-CDM, and GPSR+P2P-CD. For this experiment, we assume that a sensor node is powered by a single AA battery which has capacity of 2000mWh, which is a usual energy source for modern sensor motes. We measure the network lifetime by varying the communication radius. As shown in Figure 12, we observe slight increases in network lifetime for all three algorithms (GPSR, GPSR+P2P-CD, and GPSR+RE-CDM) as we increase the communication radius. One reason is that higher communication radius helps to increase the network lifetime by using less number of packet transmissions, but the effect of communication radius is almost negligible. Comparing network lifetime among these three protocols, we note that GPSR has the shortest lifetime; GPSR+RE-CDM has slightly better lifetime than GPSR; and GPSR+P2P-CD highly improves the network lifetime. These results conform to the

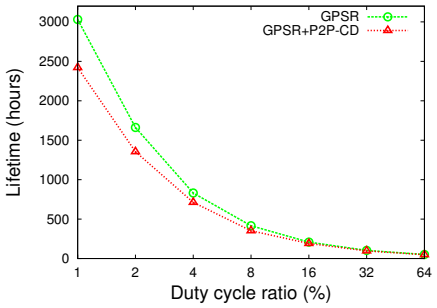

Fig. 13. Network lifetime for GPSR and GPSR+P2P-CD for different duty cycle ratios. As shown, large duty cycle ratio makes the network lifetime shorter. The gap between GPSR and GPSR+P2P in network lifetime becomes smaller as the duty cycle ratio increases.

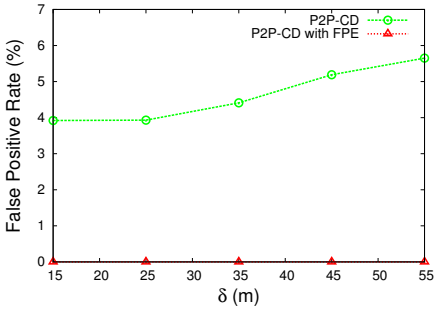

Fig. 14. Impact of parameter δ on the rate of false positives. As shown, larger δ means less precise approximation for the cut boundary, hence imprecise determination for destination reachability (i.e., some destinations may be deemed not reachable, when they are).

distribution of energy consumption in the network.

Higher duty cycle ratios expedite the aging process, because nodes have to be in the wake-up state for longer period of time. Figure 13 shows the impact of the duty cycle ratio on network lifetime. As shown, as the duty cycle ratio increases, the network lifetime rapidly decreases. One notable observation is that the gap in the network lifetime between GPSR and GPSR+P2P-CD becomes smaller with increasing duty cycle ratio. A reason is that when the duty cycle ratio is large, it is more likely that a node sends a packet to its neighbors by using the predefined working schedule, instead of using an additional wake-up slot.

5.3 False Positive Rate

A false positive occurs when a node determines that a destination is unreachable, even though the destination can actually be reached. As we described previously in Section 4.2.1, this false positive is caused by the inaccuracy in describing the boundary of a cut region; in other words, when a polygon representing the cut boundary cannot cover all the nodes in the cut region, we observe the false positives. The parameter

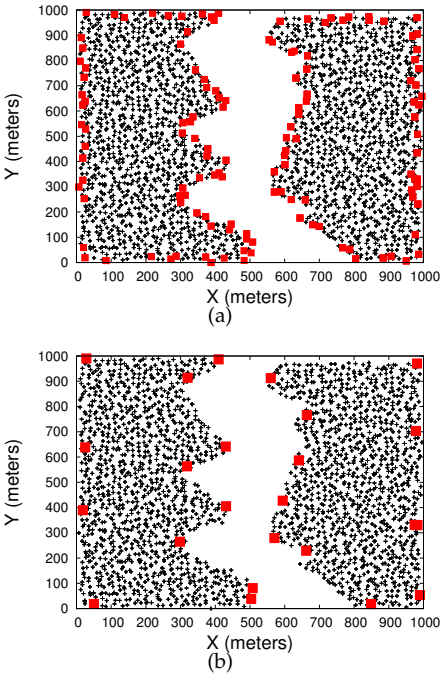

Fig. 15. (a) A cut abstracted with $\delta = 10$ meters. As shown a smaller δ allows a more precise approximation of the cut boundary. This is done, however, at the expense of more points describing the cut boundary, an overhead. (b) A cut abstracted with $\delta = 50$ meters. As shown, fewer points/rectangles are used for describing the cut boundary, at the risk of more false positives, as shown in Figure 14.

δ can be used to adjust the accuracy of the boundary description. Smaller δ values permit more precise representation of the boundary, while incurring higher overhead. We measure the false positive rate from 10,000 random source and destination pairs. Figure 14 shows the results. As expected, the false positive rate increases as the δ value increases; however, even for small δ values, the false positive persists. In Section 4.2.1, we introduced a FPE (False Positive Elimination) scheme to eliminate the false positives at the cost of storage overhead. Figure 14 depicts the results we obtained after the FPE is applied. As shown, the false positive is 0% when the FPE algorithm is applied.

5.4 Packet Size

Smaller δ values allow a precise description of a cut region, because a polygon with more vertices is used to describe the cut region. However, the size of the control packet, that is broadcast to nodes in the cut region, must be large to contain more vertices. In contrast, larger δ values permit smaller control packet size, because fewer vertices are used to represent the cut region. This, however, is done at the cost of possible errors, because the bounding box with large width may contain the nodes that do not belong to the cut region. Figure 15(a) and 15(b) show how the

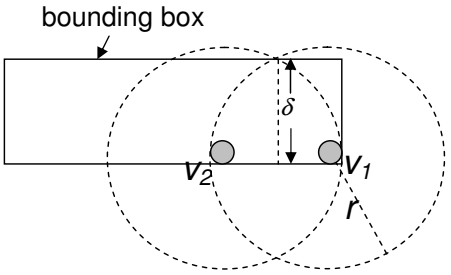

Fig. 16. An illustration of the proof for choosing optimal δ. Given any node, say v_1, in a bounding box, we attempt to find the position of its neighboring node, say v_2, such that δ, the height of the intersection area, is minimized. The δ is minimized as $\frac{\sqrt{3}}{2} \cdot r$, when v_2 is located as shown in this figure. Thus the width of a bounding box can be at most $\frac{\sqrt{3}}{2} \cdot r$; otherwise, a bounding box may contain a node that is reachable from neither v_1 nor v_2

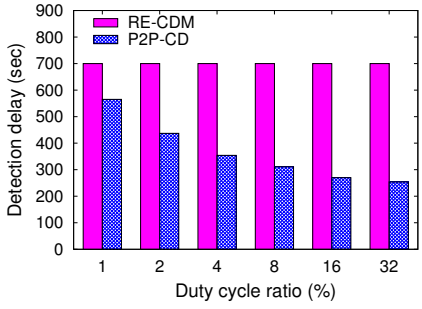

Fig. 17. Impact of duty cycle ratio on the cut detection delay. RE-CDM is not affected by duty cycle, while P2P-CD is affected by duty cycle.

vertices for the polygon are chosen for different δ values, and Table II presents the relationship between the δ values and corresponding number of vertices.

TABLE 2
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

Based on the simulation results, we attempt to determine the appropriate δ value for our experiments. We note that the largest δ value that does not cause errors (i.e., finding a bounding box that contains a vertex not within the cut boundary) is $\frac{\sqrt{3}}{2} \cdot r$, where r is the communication radius of a node (See Figure 16 for proof). Therefore, in our simulation settings, we choose $\frac{\sqrt{3}}{2} \cdot 70 \approx 60$ meters as our δ value, because the default communication radius is 70 meters.

5.5 Detection Delay

Detection delay is defined as the elapsed time between the occurrence of a cut and the detection of the cut. In RE-CDM, a node detects a cut when its state converges either to a new positive value, or 0. As Figure 17 illustrates, RE-CDM requires 7 iterations until nodes being able to detect a cut, regardless of the duty cycle ratio. Note that the iteration period of the RE-CDM was set to 100 seconds. These results indicate that if we use a smaller iteration period, RE-CDM would detect a cut faster; but, the control packet overhead would increase with the smaller iteration periods.

The P2P-CD algorithm detects a cut when the probing packet finishes traveling around the boundary of the cut region, and the set of vertices of the polygon representing the boundary is broadcast to the nodes in the cut region. Figure 17 shows the detection delay for P2P-CD per duty cycle ratio. For larger duty cycle ratio, a node has more active neighboring nodes. As a result, larger duty cycle ratio allows the probing packet to travel faster, and thus reducing the detection delay, as the figure indicates.

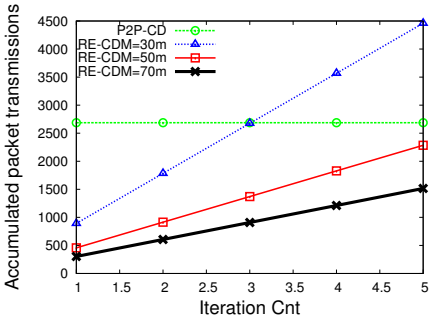

Fig. 18. Control packet overhead of P2P-CD and RECDM. A reactive solution, P2P-CD, has lower control packet overhead, compared with the RE-CDM, a proactive solution.

5.6 Control Packet Overhead

A control packet refers to a packet used to detect a cut. The P2P-CD algorithm has two types of control packets: the probing packet, and the broadcast packet used to distribute the polygon to the nodes in a cut region. In RE-CDM, a control packet is the packet that carries the state message. In this experiment, we measure the overhead of this control packet, in the form of the accumulated number of control packet transmissions. For this experiment, we fix the duty cycle ratio to 1%, and delta to 60m; and we vary communication radius and operation time.

The P2P-CD algorithm is a reactive solution; thus, once it detects a cut, it does not incur additional control message overhead, unless different cuts appear in the network. Figure 18 depicts the results. As shown, the control packet overhead for P2P-CD is constant, because the cut has been detected before 100 second (i.e., the time for the first iteration of the RE-CDM algorithm). On the other hand, the accumu-

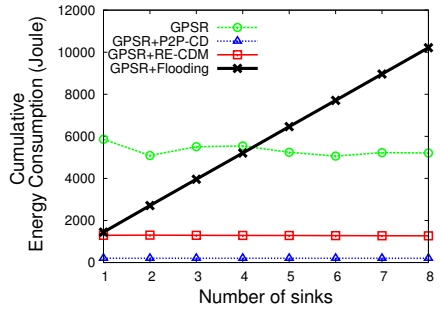

Fig. 19. Energy consumption for a network with a small number of target destinations. RE-CDM performs well in this particular network setting, compared with P2P-CD. The energy consumption of the flooding method grows fast even in the network with a few sinks.

lated number of control packet transmissions for RE-CDM continuously increases, because RE-CDM is a proactive solution that periodically scans the network for detecting a cut. Figure 18 depicts the results. In particular, when the communication radius of a node is smaller, the control packet overhead becomes higher, because a control packet must be transmitted over a larger number of hops to travel the same distance.

5.7 Number of sinks

In the previous experiments, we have focused on a scenario with N target destinations, where N is the number of nodes in the network. Now we consider a new scenario with fewer target destinations ranging from 1 to 8. Performing simulations under this new scenario is important because our RE-CDM is specifically designed for a network with a small number of target nodes (e.g., a wireless sensor network with multiple sinks). In this set of experiments, we are interested in how our RE-CDM performs in terms of energy efficiency when there are a small number of target destinations. In particular, we compare RE-CDM with the flooding method, a primitive cut detection algorithm, where each sink node broadcasts a probing packet throughout the network in order to enable nodes detect a cut. We assume that when an event occurs (i.e., every 10 seconds at a random location according to the previous experimental settings), each node reports data to a randomly selected sink node. We measure the accumulated energy consumption in Joules after 10,000 seconds of operation time.

Figure 19 depicts the results. Compared with the scenario with N target destinations, which is shown in Figure 9, RE-CDM performs significantly better, having as low energy consumption as P2P-CD. Furthermore, unlike P2P-CD, RE-CDM does not require nodes to maintain global topology information in their storages, nor the location information of neighboring nodes. Note that, as RE-CDM does not require location information, it can be coupled with routing protocols that are not based on node localization. Although the energy consumption of RE-CDM would gradually increase as operation time elapses due to the periodic state-message exchanges, we believe that it is a proper solution for the applications where nodes have limited storage and computational capabilities, and appropriate localization methods are unknown.

Investigating the energy consumption of the flooding method, the simplest cut detection scheme, allows us to understand where our proposed solutions stand in terms of energy efficiency. As shown in Figure 19, the flooding method shows as low energy consumption as RE-CDM when there is a single target destination (i.e., a sink). However, if we increase the number of sink nodes, consumed energy for the flooding method linearly increases, because each sink node periodically broadcasts a probing packet throughout the network. For more than 5 sink nodes, the flooding method performs even worse than when no cut detection algorithm is used (i.e., the GPSR). Furthermore, the flooding method suffers from the limitations that only a part of nodes is able to detect a cut; thus, this method can only be useful for a network with very small number of target destinations (e.g., a single sink node), where only the nodes in a disconnected network segment are required to detect a cut.

6 Conclusions

In this article, we introduced a new problem, called *the destination-based cut detection*, which extends the notion of the existing cut detection problem. This new problem was derived from a novel taxonomy for cut detection schemes; systematically organizing existing cut detection algorithms is expected to provide guidelines for future research on this topic, as well as improving the understanding of our contributions.

We then proposed two algorithms to address the destination-based cut detection problem. We first introduced the point-to-point cut detection protocol (P2P-CD) which enables each node to be able to detect a cut with respect to any destination node. This protocol significantly reduces energy consumption when coupled with an underlying routing protocol at the cost of the knowledge on partial global topology. Our second algorithm, the robust and energy efficient cut detection for multiple sinks (RE-CDM) is a more lightweight solution in that it does not require information on global topology, nor node's location information. This algorithm was designed for network scenarios with a small number of sink nodes. Through extensive simulations, we showed that both algorithms achieve more than an order of magnitude improvement in energy consumption, when coupled with an underlying routing protocol.

REFERENCES

[1] S. M. George, W. Zhou, H. Chenji, M. Won, Y. Lee, A. Pazarloglou, R. Stoleru, and P. Barooah, "DistressNet: a wireless AdHoc 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 *Proceedings of ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2004.

[6] H. Ritter, R. Winter, and J. Schiller, "A partition detection system for mobile ad-hoc networks," in *Proceedings of IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (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 *Proceedings of IEEE Conference on Decision and Control and European Control Conference (CDC)*, 2008.

[10] P. Barooah, H. Chenji, R. Stoleru, and T. Kalmar-Nagy, "Cut detection in wireless sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 99, no. PrePrints, 2011.

[11] E. Oyman and C. Ersoy, "Multiple sink network design problem in large scale wireless sensor networks," in *Proceedings of IEEE International Conference on Communications (ICC)*, 2004.

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

[13] V. Park and M. Corson, "A highly adaptive distributed routing algorithm for mobile wireless networks," in *Proceedings of IEEE International Conference on Computer Communications*, 1997.

[14] 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.

[15] 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.

[16] M. Won, M. George, and R. Stoleru, "RE^2-CD: Robust and energy efficient cut detection in wireless sensor networks," in *Proceedings of International Conference on Wireless Algorithms, Systems, and Applications (WASA)*, 2009.

[17] M. Won and R. Stoleru, "Destination-based cut detection in wireless sensor networks," in *Proceedings of IEEE/IFIP International Conference on Embedded and Ubiquitous Computing (EUC)*, 2011.

[18] 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.

[19] B. Karp and H. T. Kung, "GPSR: Greedy perimeter stateless routing for wireless networks," in *Proceedings of ACM International Conference on Mobile Computing and Networking (MOBICOM)*, 2000, pp. 243–254.

[20] G. Tan, M. Bertier, and A.-M. Kermarrec, "Visibility-graph-based shortest-path geographic routing in sensor networks," in *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, 2009.

[21] 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.

[22] L. Tang, Y. Sun, O. Gurewitz, and D. Johnson, "Pw-mac: An energy-efficient predictive-wakeup mac protocol for wireless sensor networks," in *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, 2011.

[23] M. Maróti, B. Kusy, G. Simon, and A. Lédeczi, "The flooding time synchronization protocol," in *Proc. of ACM SenSys*, 2004.

[24] *CC2420 Data Sheet, http://www.chipcon.com*.

PLACE PHOTO HERE

Myounggyu Won received his B.E. degree with honors from Sogang University, Seoul, Korea. He is currently pursuing his Ph.D. degree in the Department of Computer Science and Engineering at Texas A&M University. His main research interests include designing energy-efficient protocols for wireless sensor networks with complex network topologies such as network cuts and holes.

PLACE PHOTO HERE

Radu Stoleru is an Assistant Professor in the Department of Computer Science and Engineering at Texas A&M University and the head of Laboratory for Embedded & Networked Sensor Systems (LENSS). His research interests are deeply embedded wireless sensor systems, distributed systems, embedded computing, and computer networking. He has authored over 60 papers and won the Outstanding Graduate Student Research Award from the Department of Computer Science, University of Virginia in 2007. He received a Ph.D. in Computer Science from the University of Virginia in 2007.