

## **A New Distributed Cut Detection Approach for Identifying Cuts in Wireless Sensor Networks**

Shiva Kumar Rameswarapu<sup>#1</sup>, Dr.M.V.Rama Sundari<sup>\*2</sup>, Dr.S.Maruthu Perumal<sup>\*3</sup>

<sup>#1</sup>IInd M.TECH (CSE) Student, <sup>\*2</sup> Associate Professor, <sup>\*3</sup> Professor & HOD  
Department of CSE  
Godavari Institute of Engineering and Technology (GIET),  
Rajahmundry, AP, India.

### **Abstract**

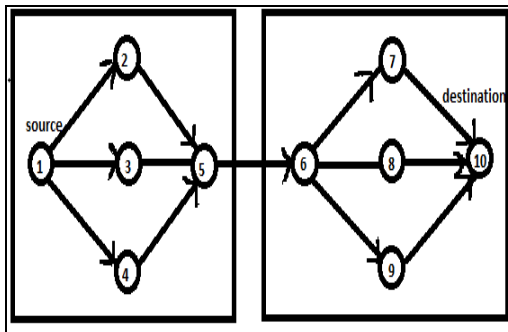
A wireless sensor network consists of spatially distributed autonomous sensors to physical or environmental conditions, such as temperature, sound, pressure, etc. and to cooperatively pass their data through the network to main location. When the data transmission between one node to another, there is some of the nodes in the network are failed, because of above mentioned issues. So, the data will not be transferred to the destination. Because of these failures of nodes the network is divided into multiple parts. The ability to detect the cuts by using the source node and disconnected node of a wireless sensor network will lead to the increase in the lifetime of network. In this article we consider the problem of detecting cuts by the remaining nodes of a wireless sensor network. We propose an algorithm that allows (i) every node to detect when the connectivity to a specially designated node has been lost, and (ii) one or more nodes (that are connected to the special node after the cut) to detect the occurrence of the cut. The algorithm is distributed and asynchronous: every node needs to communicate with only those nodes that are within its communication range. The algorithm is based on the iterative computation of a fictitious “electrical potential” of the nodes. The convergence rate of the underlying iterative scheme is independent of the size and structure of the network. We demonstrate the effectiveness of the proposed algorithm through simulations and a real hardware implementation.

### **Keywords:**

Wireless networks, sensor networks, network separation, detection and estimation, iterative computation

### **1. Introduction**

A wireless sensor network is a collection of nodes organized into a network such that each node having sensing and processing capabilities. Each node has an RF transceiver, sensor, memory, powered by battery. Nowadays sensors are widely employed in various research fields since they can monitor temperature and hence whether forecasting can be made easier. They are randomly deployed in areas with sensors attached according to the applications for which they are being used. Since they are being powered up by batteries, energy consumption should be minimized in order to prolong the life of sensor nodes. In a network, sensor nodes communicate with each other so that results are obtained as part of their cooperatively combined work. Since each node needs to communicate with all the other nodes, wireless links are established between them. A cut is defined as the failure of node. It can separate the network into disconnected paths incapable of communicating with each other. Since they are randomly deployed, loss of connectivity can be quite disastrous as they will lead to the breakdown of entire network.



**Fig. 1 A wireless sensor network showing network connectivity and cut vertices**

Wireless sensor networks (WSNs) are a promising technology for monitoring large regions at high spatial and temporal resolution. However, the small size and low cost of the nodes that makes them attractive for widespread deployment also causes the disadvantage of low operational reliability. The node may fail due to different problems. In fact, node failure is expected to be quite common due to the typically limited energy budget of the nodes that are powered by small batteries. Failure of a set of nodes will reduce the number of multi hop paths in the network. Such failures cause a subset of nodes—that have not failed—to become disconnected from the rest, resulting in a “cut”.

We consider the problem of detecting cuts by the nodes of a wireless sensor network. May source node is a base station serves as an interface between the network and its users. So, cut may or may not separate a node from the source node, when a node is disconnected from the source is  $u$ , when a cut occurs in the network that does not separate a node  $u$  from the source node, we say that these nodes are connected, but a cut occurred somewhere (CCOS) event has occurred for  $u$ . By cut detection we mean 1) detection by each node of DOS event when it occurs, and 2) detection of CCOS events by the nodes close to a cut, and the approximate location of the cut.

Nodes that detect the occurrence and approximation locations of the cuts can then alert the source node or the base station. if a node having the

ability to detect the cut, it could simply wait for the network to be repaired and eventually reconnected, so it saves the energy of the multiple nodes after cut.

To see the benefits of a cut detection capability, imagine that a sensor that wants to send data to the source node has been disconnected from the source node. Without the knowledge of the network’s disconnected state, it may simply forward the data to the next node in the routing tree, which will do the same to its next node, and so on. However, this message passing merely wastes precious energy of the nodes; the cut prevents the data from reaching the destination. Therefore, on one hand, if a node were able to detect the occurrence of a cut, it could simply wait for the network to be repaired and eventually reconnected, which saves onboard energy of multiple nodes and prolongs their lives.

On the other hand, the ability of the source node to detect the occurrence and location of a cut will allow it to undertake network repair. Thus, the ability to detect cuts by both the disconnected nodes and the source node will lead to the increase in the operational lifetime of the network as a whole. A method of repairing a disconnected network by using mobile nodes has been proposed in Algorithms for detecting cuts, as the one proposed here, can serve as useful tools for such network [1].  $k$  repairing methods. A review of prior work on cut detection in sensor networks, e.g. [2], [3], [4] and others, is included in the Supplementary Material.

In this paper we propose a distributed algorithm to detect cuts, named the Distributed Cut Detection (DCD) algorithm. The algorithm allows each node to detect DOS events and a subset of nodes to detect CCOS events. The algorithm we propose is distributed and asynchronous: it involves only local communication between nodes, and is robust to temporary communication failure between node pairs. A key component of the DCD algorithm is a distributed iterative computational step through which nodes compute their electrical potentials. The convergence rate of the computation is independent of the size and structure of the network.

## 2. Problem Statement

Consider a sensor network modeled as a undirected graph  $G=(V,E)$ , whose node set  $V$  represents the sensor nodes and the edge set  $E$  consists of pair of nodes  $(u, v)$  such that nodes  $u$  and  $v$  can exchange messages between each other. Note that we assume inter-node communication is symmetric. An edge  $(u, v)$  is said to be incident on both the  $u$  and  $v$ . The nodes that share an edge with a particular node  $u$  are called the neighbors of  $u$ . A cut is the failure of a set of nodes  $V_{cut}$  from  $G$  results in  $G$  being divided into multiple connected components [6]. Recall that an undirected graph is said to be connected if there is a way to go from every node to every other node by traversing the edges, and that a component  $G_c$  of a graph  $G$  is a maximal connected sub graph of  $G$ . We are interested in devising a way to detect if a subset of the nodes has been disconnected from a distinguished node, which we call the source node, due to the occurrence of a cut.

## 3. Distributed Cut Detection

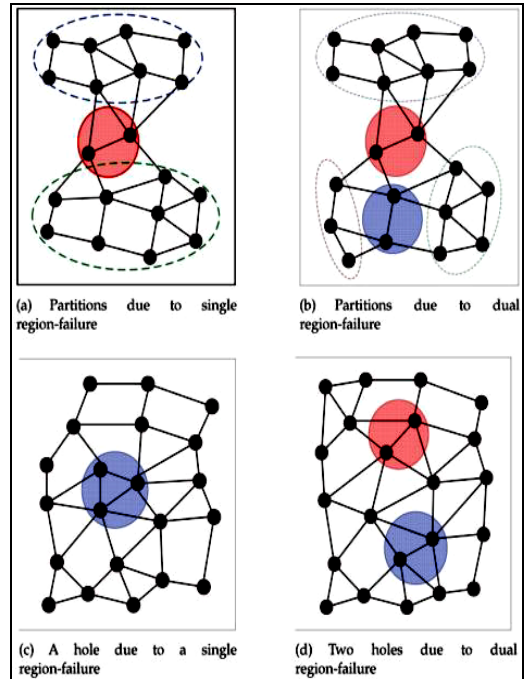
The algorithm is based on an electrical analogy. Given an undirected graph  $G = (V, E)$  with, say,  $n$  nodes  $m$  edges that describes the sensor network, this algorithm is used for the nodes which is disconnected from the source node. We construct the graph  $G^{elec} = (V^{elec}, E^{elec})$  where  $V^{elec} = V \cup V^{fict}$ , where  $V^{fict}$  consists of  $n - 1$  nodes, one node for every node in  $V$  except the source node, and  $V$  is connected to it fictitious node in  $V^{fict}$  with a single edge.

In this algorithm we are having two phases. One is state update law, it works very efficient to calculate the node potentials in electrical network (Gelec,1) when  $s$  Ampere current is injected to the source node and extracted to the nodes  $V^{fict}$ , with all nodes in  $V$ . The other phase of the algorithm consists of monitoring the state of a node, it is used to detect the cut occurred. Now we describe about the each phase.

### I. State Update Law

The nodes use the computed potentials to detect if DOS events have occurred (ie. if they are

disconnected from the source node). To detect CCOS events, the algorithm uses the fact that the potentials of the nodes that are connected to the source node also change after the cut. CCOS detection proceeds by using probe messages that are initiated by certain nodes that encounter failed neighbors, if a short path exists around a “hole” created by node failures, the message will reach the initiating node.



**Fig. 2. A graph describing a sensor network and associated electrical network.**

Every node keeps a scalar variable, which is called a state Let  $G(k) = (V(k), E(k))$  represent the sensor network that consists of all the nodes and edges of  $G$  that are still active at time  $k$ , where  $k=|0, 1, 2, \dots$  Is an iteration counter. We index the source node as 1. Every node  $u$  maintains a scalar state  $x_u(k)$  that is updated. At every iteration  $k$ , nodes broadcast their current states. Let  $N_u(k) = \{v|(u,v) \in E(k)\}$  denote the set of neighbors of  $u$  in the graph  $G(k)$ . Every node in  $V$  except the source node

updates the following state law. Where strength is design parameter:

$$X_i(k+1) = \frac{1}{D_i(k) + 1} \sum_{j \in N_i(k)} x_j(k) + s1_{\{i\}} \quad (1)$$

Where  $d_i(k) := |N_i(k)|$  is the degree of node  $i$  at time  $k$ , and  $1_A(i)$  is the indicator function of the set  $A$ . That is,  $1_{\{i\}}(i) = 1$  if  $i=1$ , and  $1_{\{i\}}(i) = 0$  if  $i \neq 1$ . After that,  $i$  can update its neighbor list  $N_i(k)$  as follows: if no messages have been received from a neighboring node for the past  $T_{drop}$  iterations, node  $i$  drops that node from list of neighbors. The integer parameter  $T_{drop}$  is a design choice.

When the network is connected, the state of a node converges to its potential in the electrical network  $(G^{elec}, 1)$ , which is a positive number. The potential of a node that is disconnected from the source is 0; this is the value converges to 0. If the reconnection occurs after a cut, the states of reconnected nodes again converge to positive values. Especially with wireless communication an asynchronous update is preferable.

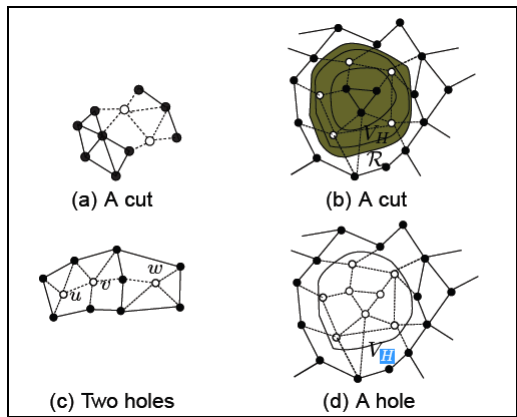


Fig. 3. Examples of cuts and holes. Filled circles represent active nodes and unfilled filled circles represent failed nodes. Solid lines represent edges, and dashed lines represent edges that existed before the failure of the nodes. The hole in (d) is indistinguishable from the cut in (b) to nodes that lie outside the region  $R$ .

## II. State monitoring for cut detection

Theorem 1 shows how the occurrence of a cut in the Network is manifested in the states of the nodes. By analyzing their own states, nodes can detect if a cut has occurred.

Suppose a cut occurs at some time  $\tau > 0$  which separates the network into  $n$  components  $G_{source}, G_2, \dots, G_n$ , the component  $G_{source}$  containing the source node. Since there is no source (and therefore no current injection) in each of the components  $G_2, \dots, G_n$  disconnected from the source, it follows from Theorem 1 that the state of every node in each of these components will converge to zero. When the potential at a particular node drops below a particular threshold value, the node can declare itself cut from the source node. In fact, there may be additional node failures (and even increase in the number of components) after the cut appears. Since the state of a node converges to 0 if there is no path to the source, additional time variation in the network will not affect cut detection. If additional failures do not occur after the cut occurs, it follows from Theorem 1 that the states of the nodes that are in the component  $G_{source}$  (which contains the source) will converge to new steady state values. So, if a node detects that its state has converged to a steady state, then changed, and then again converged to a new steady state value that is different from the initially seen steady state, it concludes that there has been a cut somewhere in the network. A node detects when steady state is reached by comparing the derivative of its state (with respect to time) with a small number  $\epsilon$  that is provided a-priori. The parameters  $s$  and  $\epsilon$  are design variables. It updates its state from that neighbor, in the asynchronous setting every node keeps a local iteration counter that may differ from those of other nodes by arbitrary amount.

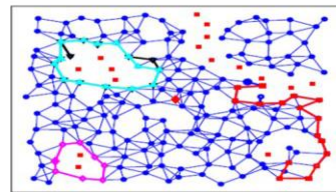


Fig. 4.  $G(k)$  for  $k > 100$

The source node is at the center. The nodes 3b fail at  $k = 100$ , and thereafter they do not participate in communication or computation. Figs. The state of a node  $u$  decays to 0 after reaching a positive value, where the state of node  $v$  says positive.

#### 4. Distributed Cut Detection Algorithm

The Distributed Cut Detection Algorithm is shown as follows:

##### A) Dos Detection

We say that a Disconnected from Source (DOS) event has occurred for  $u$ . The algorithm allows each node to detect DOS events. The nodes use the computed potentials to detect if DOS events have occurred (i.e., if they are disconnected from the source node). The approach here is to exploit the fact that if the state is close to 0 then the node is disconnected from the source, otherwise not. In order to reduce sensitivity of the algorithm to variations in network size and structure, we use a normalized state. DOS detection part consists of steady-state detection, normalized state computation, and connection/separation detection. A node keeps track of the positive steady states seen in the past using the following method. Each node  $i$  computes the normalized state difference  $\delta x_i(K)$  as follows:

$$\delta x_i(k) = \begin{cases} \frac{x_i(k) - x_i(k-1)}{X_i(k-1)}, & \text{if } x_i(k-1) > \epsilon_{zero} \\ \infty, & \text{otherwise,} \end{cases}$$

Where  $\epsilon_{zero}$  is a small positive number. A node  $i$  keeps a Boolean variable Positive Steady State Reached (PSSR) and updates  $PSSR(k) \leftarrow 1$  if  $|\delta x_i(K)| < \epsilon_{\Delta x}$  for  $K = k - T_{guard,k} - T_{guard} + 1, \dots, k$  (i.e., for  $T_{guard}$  consecutive iterations), where  $\epsilon_{\Delta x}$  is a small positive number and  $T_{guard}$  is a Small integer. The initial 0 value of the state is not considered a steady state, so  $PSSR(k) = 0$  for  $k = 0, 1, \dots, T_{guard}$ .

##### B) CCOS Detection

The algorithm for detecting CCOS events relies on finding a short path around a hole, if it exists, and is partially inspired by the jamming detection algorithm proposed in [5]. When a cut occurs in the network that does not separate a node  $u$  from the source node, we say that Connected, but a Cut Occurred Somewhere (CCOS) event has occurred for  $u$ . detection of CCOS events by the nodes close to a cut, and the approximate location of the cut. By “approximate location” of a cut we mean the location of one or more active nodes that lie at the boundary of the cut and that are connected to the source. To detect CCOS events, the algorithm uses the fact that the potentials of the nodes that are connected to the source node also change after the cut. However, a change in a node’s potential is not enough to detect CCOS events, since failure of nodes that do not cause a cut also leads to changes in the potentials of their neighbors. Therefore, CCOS detection proceeds by using probe messages.

#### 5. System Implementation

In this section, we describe the software implementation and evaluation of the DCD algorithm. In software the java language algorithm was implemented using the java language running on windows xp operating system. The system executes in two phases: the Reliable Neighbor Discovery (RND) phase and the DCD Algorithm phase. In the RND phase each node is connected to the source node. Upon receiving the message, the mote updates the number of beacons received from that particular sender.

To determine whether a communication link is established, each mote first computes for each of its neighbors the Packet Reception Ratio (PRR), defined as the ratio of the number of successfully received beacons and the total number of beacons sent by a neighbor. A neighbor is deemed reliable if the  $PRR > 0.8$ . Next, the DCD algorithm executes. After receiving state information from neighbors, a node updates its state according to (1) in an asynchronous manner and broadcasts its new state. The state is stored in the database.

## 6. Conclusion

The DCD algorithm we propose here enables every node of a wireless sensor network to detect Disconnected from Source events if they occur. Second, it enables a subset of nodes that experience CCOS events to detect them and estimate the approximate location of the cut in the form of a list of active nodes that lie at the boundary of the cut/hole. The DOS and CCOS events are defined with respect to a specially designated source node. The algorithm works effectively with large classes of graphs of varying size and structure, without requiring changes in the parameters. For certain scenarios, the algorithm is assured to detect connection and disconnection to the source node without error. A key strength of the DCD algorithm is that the convergence rate of the underlying iterative scheme is quite fast and independent of the size and structure of the network, which makes detection using this algorithm quite fast application of the DCD algorithm to detect node separation and reconnection to the source in mobile networks is a topic of ongoing research.

## 7. References

- [1] G. Dini, M. Pelagatti, and I. M. Savino, "An algorithm for reconnecting wireless sensor network partitions," in *European Conference on Wireless Sensor Networks*, 2008, pp. 253–267.
- [2] N. Shrivastava, S. Suri, and C. D. T'oth, "Detecting cuts in sensor networks," *ACM Trans. Sen. Netw.*, vol. 4, no. 2, pp. 1–25, 2008.
- [3] H. Ritter, R. Winter, and J. Schiller, "A partition detection system for mobile ad-hoc networks," in *First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (IEEE SECON 2004)*, Oct. 2004, pp. 489–497.
- [4] M. Hauspie, J. Carle, and D. Simplot, "Partition detection in mobile ad-hoc networks," in *2nd Mediterranean Workshop on Ad-Hoc Networks*, 2003, pp. 25–27.
- [5] A. D. Wood, J. A. Stankovic, and S. H. Son, "Jam: A jammed-area mapping service for sensor networks," in *IEEE Real Time System Symposium*, 2003.
- [6] P. Barooah, "Distributed cut detection in sensor networks," in *47<sup>th</sup> IEEE Conference on Decision and Control*, December 2008, pp. 1097 – 1102.
- [7] A.D. Wood, J.A. Stankovic, and S.H. Son, "Jam: A Jammed-Area Mapping Service for Sensor Networks," *Proc. IEEE Real Time Systems Symp.*, 2003. International Journal of Engineering Research & Technology (IJERT) Vol. 1 Issue 10, December- 2012 ISSN: 2278-0181.
- [8] G. H. Golub and C. F. van Loan, *Matrix Computations*, 3rd ed. The John Hopkins University Press, 1996.
- [9] F. Chung, "Spectral graph theory," Regional Conference Series in Mathematics, Providence, R.I., 1997.
- [10] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest Neighbor rules," *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 988–1001, June 2003.

## 8. About the Authors

**Shiva Kumar Rameswarapu** is currently pursuing his M.Tech (CSE) in Computer Science & Engineering Department, GIET, Rajahmundry. His area of interests includes Networks

**Dr.M.V.Rama Sundari** is currently working as an Associate Professor in Department of IT, GIET, Rajahmundry. She was awarded with PhD in related field. Her research interests include Communications Networks.

**Dr. S. Maruthu Perumal** is currently working as a Head of the Department for Computer Science & Engineering Department, GIET, Rajahmundry. He is awarded with PhD in related field. His research interests include Image Processing, Data Mining & Warehousing, Networks and Security, Software Engineering.