Volume 3, Issue 11, November 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Detecting Disconnection in Wireless Sensor Network

Lagdive Kedar.M.[1]
Computer Science, & Engg.
India

Sudhir S. Kanade[2],
Department of ETC COE Osmanabad
India

Abstract*— In a Wireless Sensor Network, sensor nodes may fail for several reasons and the network may split into two or more disconnected partitions. This may deteriorate or even nullify the usefulness and effectiveness of the network. 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 small set of target destinations. The convergence rate of the underlying iterative scheme is independent of the size and structure of the network.

Keywords-Wireless sensor networks, network separation, monitoring, Detection and Estimation, Iterative computation

I] INTRODUCTION

Sensor networks offer a powerful combination of distributed sensing, computing and communication. They lend themselves to countless applications and, at the same time, offer numerous challenges due to their peculiarities, primarily the stringent energy constraints to which sensing nodes are typically subjected. The increasing interest in wireless sensor networks can be promptly understood simply by thinking about what they essentially are: a large number of small sensing self-powered nodes which gather information or detect special events and communicate in a wireless fashion, with the end goal of handing their processed data to a base station. A node may fail due to various factors such as mechanical/electrical problems, environmental degradation, battery depletion, or hostile tampering. 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 can cause a subset of nodes – that have not failed – to become disconnected from the rest, resulting in a "cut". Two nodes are said to be disconnected if there is no path between them[1].

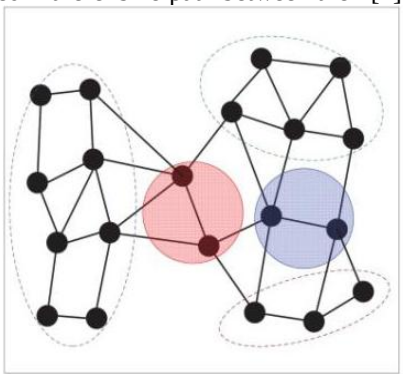

Fig. 1 Partition due to two region failure

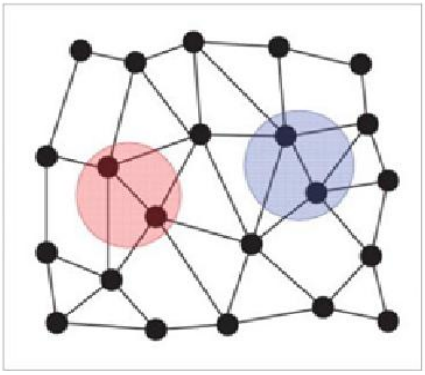

Fig. 2 Partition due to two holes in network

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.

Manuscript received 2011 IFIP Ninth International Conference on Embedded and Ubiquitous Computing.This work is supported by Dr. Babasaheb A. M. University, Lagdive K.M. [M.E. II] is from Computer Engineering TPCTs COE Osmanabad.

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. In our view, power efficiency, scalability most important considerations for a scheme to detect network cuts.

II] PROBLEM STATEMENT

According to the tunnel-disaster-scenario, the wireless sensor network is composed of a powerful base station and a set of low-end sensor nodes. Base station and sensor nodes have wireless capabilities and communicate through a wireless, multi-hop, ad-hoc network. We assume the resulting wireless network runs a routing algorithm that is able to cope with the failure of a "small" number of nodes by finding alternative routes .However, in the case of a disaster, the number of failed nodes is "too large" and the network breaks into two or more disconnected partitions.

Consider a sensor network modelled 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 neighbours 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. 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 GC 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.

III] DETECTION OF DISCONNECTION

The problem we seek to address is twofold. First, we want to enable every node to detect if it is disconnected from the source (i.e., if a DOS event has occurred). Second, we want to enable nodes that lie close to the cuts but are still connected to the source (i.e., those that experience CCOS events) to detect CCOS events and alert the source node. The DCD algorithm is based on the following electrical analogy. Imagine the wireless sensor network as an electrical circuit where current is injected at the source node and extracted out of a common fictitious node that is connected to every node of the sensor network. Each edge is replaced by a 1 resistor. When a cut separates certain nodes from the source node, the potential of each of those nodes becomes 0, since there is no current injection into their component. The potentials are computed by an iterative scheme (described in the sequel) which only requires periodic communication among neighbouring nodes. The nodes use the computed potentials to detect if DOS events have occurred (i.e., 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. 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 neighbours. Therefore, CCOS detection proceeds by using probe messages that are initiated by certain nodes that encounter failed neighbours, and are forwarded from one node to another in a way that if a short path exists around a "hole" created by node failures, the message will reach the initiating node. The nodes that detect CCOS events then alert the source node about the cut(refer the figure 1 and 2).

IV] PHASE OF STATE UPDATING

Every node keeps a scalar variable, which is called its *state*. 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,…. is an iteration counter. We index the source node as 1. Every node u maintains a scalar state xu(k) that is updated. At every iteration k, nodes broadcast their current states. Let Nu(k) = {v|(u,v) € 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:

$$Xi(k+1) = \frac{1}{Di(k)+1} \sum_{j \in Ni(k} xj(k) + s1\{1\} \quad (i)$$

Where di(k) := | Ni(k) | is the degree of node i at time k, and 1A(i) is the indicator function of the set A. That is, 1{1} (i) = 1 if i=1, and 1{1}(i) = 0 if i=1 and. After that, i can update its neighbour list Ni(k) as follows: if no messages have been received from a neighbouring node for the past T drop iterations[2],[3] node i drops that node from list of neighbours[5],[6]. When the sensor network is connected, the state of a node converges to its potential in the electrical network which is a positive number. If a cut occurs, the potential of a node that is disconnected from the source is 0; and this is the value its state converges to. If reconnection occurs after a cut, the states of reconnected nodes again converge to positive values. Therefore, a node can monitor whether it is connected or separated from the source by examining its state. The above description assumes that all updates are done synchronously. In practice, especially with wireless communication, an asynchronous update is preferable. The algorithm can be easily extended to asynchronous setting by

letting every node keep a buffer of the last received states of its neighbours. If a node does not receive messages from a neighbour during the interval between two iterations, it updates its state using the last successfully received state from that neighbour.

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 when s Ampere current is injected to the source node and extracted to the nodes Vfict, 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.

V] ALGORITHM FOR DETECTION OF DISCONNECTION

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 as shown in fig.3. Each node computes the normalized state difference () as follows:

$$xi(k) = \frac{xi(k) - xi(k-1)}{xi(k-1)}, \quad \text{if} \quad xi(k-1) > € \quad \text{zero}$$
$$\infty \quad \text{otherwise}$$

Where, zero is a small positive number. A node keeps a Boolean variable Positive Steady State Reached (PSSR) and updates PSSR(k) k ,Tguard +1,…,k(i.e., for Tguard consecutive iterations),where is a small positive number and Tguard is a Small integer. The initial 0 value of the state is not considered a steady state, so PSSR()=0 for =0,1, … ,Tguard[6][7]. Each node keeps an estimate of the most recent ―steady state‖ observed, which is denoted by ( ). This estimate is updated at every time k. Every node i also keeps a list of steady states seen in the past, one value for each unpunctuated interval of time during which the state was detected to be steady. This information is kept in a vector ( ), which is initialized to be empty and is updated as follows: If PSSR () = 1 but PSSR(−1) = 0, then is appended to ( ) as a new entry. If steady state reached was detected in both and - 1 (i.e., PSSR( ) = PSSR(− 1) = 1, then the last entry of ( ) is updated to ( ) .

B. CCOS Detection

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[4]. 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 neighbours. Therefore, CCOS detection proceeds by using probe messages.

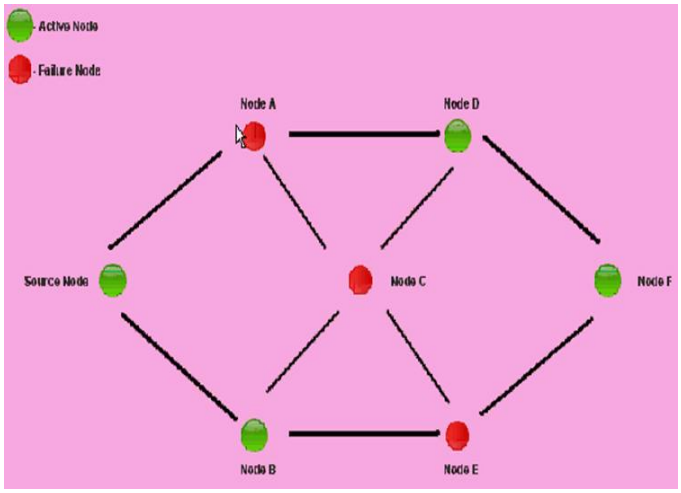

Fig. 3 detecting disconnection in network

VI] IMPLEMENTATION

In this section, we describe the software implementation and evaluation of the DCD algorithm. In software the algorithm was implemented using the java language running on windows xp operating system. The system executes in two phases: the Reliable Neighbour 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 neighbour. A neighbour is deemed reliable if the PRR >0:8. Next, the DCD algorithm executes. After receiving state information from neighbours, a node updates its state according to (1) in an asynchronous manner and broadcasts its new state.

VII] CONCLUSION

The DCD algorithm we propose here enables every node of a wireless sensor network to detect DOS (Disconnected from Source) events if they occur. Second, it enables a subset of nodes that experience CCOS (Connected, but Cut Occurred Somewhere) 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 is based on ideas from electrical network theory and parallel iterative solution of linear equations.

ACKNOWLEDGEMENT

The author is sincerely thankful to all professors for their guidance for his paper. Without his help it was tough job for him. He is really thankful for their relevant help and providing the necessary guidance given by Prof. Sudhir S. Kanade.

REFERENCES

1] 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.
2] H. Ritter, R. Winter, and J. Schiller, ―A Partition Detection System for Mobile Ad-hoc Networks, Proc. First Ann. IEEE Comm. Soc. Conf. Sensor and Ad Hoc Comm.and Networks (IEEE SECON '04), pp. 489-497, Oct. 2004.
3] P. Barooah, ―Distributed Cut Detection in Sensor Networks, Proc. 47th IEEE Conf. Decision and Control,pp. 1097-1102, Dec.2008.
4] Jadbabaie, J. Lin, and A. S. Morse, ―Coordination of groups of mobile autonomous agents using nearest neighbour rules, IEEE Transactions on Automatic Control, vol. 48, no. 6, pp. 988–1001, June 2003.
5] G.Dini, M. Pelagatti, and I.M. Savino, ―An Algorithm for Reconnecting Wireless Sensor Network Partitions, Proc. European Conf. Wireless Sensor Networks, pp. 253-267, 2008.
6] 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.
7] J. Kleinberg, "Detecting a network failure," *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, p. 231, 2000.65

