A Study Of Distributed Source Separation Detection And Energy Efficient Cut Detection Algorithms Based On An Electrical Analogy

Y. Mamatha[1], N. Krishna Vardhan[2]
[1]M.Tech(CSE), Sri Kottam Tulasi Reddy Memorial College of Engineering, Kondair, India.
[2]Asso.Professor, Sri Kottam Tulasi Reddy Memorial College of Engineering, Kondair, India.

Abstract

Wireless Sensor Networks can be used to monitor and collect various physical attributes within a specific area of interest. In most of the cases, sensor nodes may fail and the network split into two or more disconnected partitions. This may lead to an issue of effectiveness of the network. Therefore, repairing partitions is a priority. By reasoning upon the degree of connectivity with neighbors, a mobile node finds the proper position where to stop in order to re-establish connectivity. In this paper we present a method to repair network partitions by using mobile nodes.

In this paper, we study how to monitor the sensor network itself, and how to detect when the network has suffered a significant "cut"? We also study a distributed algorithm to detect "cuts" in sensor networks.

1. Introduction

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. 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.

Wireless Sensor Networks (WSNs) have been developed and extensively applied in monitoring. WSNs can be used to monitor and collect various physical attributes within a specific area or environment of interest. Therefore, WSNs can be viewed as a large database whose data readings from the sensors may be abnormal due to faulty sensors or unusual phenomenon in the monitored domain. However, with huge amount data, much energy is wasted in transmitting all of the measured data to the base station. Hence, in order to reduce energy consumption of transmitting all data should be preprocessed prior to transmission while still maintaining the acceptable anomaly detection rate. A rich variety of scientific, commercial, and military applications has been proposed for sensor networks, and many experimental prototypes are under development in academia and industry. Realizing the full potential of the sensor networks, however, requires solving several challenging research problems. Many of these challenges stem from two major limitations of the sensor nodes: low power and low bandwidth. Consequently, a number of proposals have been made for improving the data collection and information processing in sensor networks, including power-aware routing and scheduling, in network aggregation, query processing, data storage management, etc.

After all, if sensor networks are to act as our remote "eyes and ears," then we need to ensure that any significant failure (natural or adversarial) suffered by the network is promptly and efficiently detected. Tracking the operational health of the infrastructure is important in any communication network, but it is especially important in sensor networks due to their unique characteristics, and the need to perform this duty with very little overhead.

In our view, power efficiency, scalability, and absence of false positives are the three most important considerations for a scheme to detect network cuts. Because a sensor network's lifetime is largely determined by how well it conserves power, solutions where all sensors are continuously monitored are both inefficient and unscalable. Because sensor networks can vary in size from few hundred nodes to hundreds of thousands, it is also desirable to design schemes that are highly scalable, so that the task of cut detection does not end up consuming a large part of the network resources. Finally, because many sensor network applications envision unmanned and remote deployment, failure detection schemes that yield false positive, or false negatives, are highly undesirable.

We propose a distributed algorithm to detect "cuts" in sensor networks, i.e., the failure of a set of nodes that separates the networks into two or

more components. The algorithm consists of a simple iterative scheme in which every node updates a scalar state by communicating with its nearest neighbors. In the absence of cuts, the states converge to values that are equal to potentials in a fictitious electrical network. When a set of nodes gets separated from a special node, that we call a "source node", their states converge to 0 because "current is extracted" from the component but none is injected. These trends are used by every node to detect if a cut has occurred that has rendered it disconnected from the source. Although the algorithm is iterative and involves only local communication, its convergence rate is quite fast and is independent of the size of the network.

2. Network Failure Detection

In this section we identify various aspects to detect the network failures by monitoring network connectivity and detect the failure edges like monitoring network connectivity, detecting sets for edge failures and detecting sets for node failures.

Geometric Preliminaries

The network topology and the communication protocol are not directly relevant to our result. We simply assume that the sensor network is connected and that every sensor is able to communicate with a *base station* through multi-hop routing, as long as a valid communication path exists. We also assume that the location of every sensor is available to the base station. A set *S* of *n* sensors scattered in a terrain is modeled as a set of *n* points in the plane (ignoring the altitude of each sensor). Our problem of monitoring the integrity of the sensor field is best studied in a geometric setting.

a. Sentinel sets
b. A Duality Transform
c. Line Arrangements and Levels
d. Minimum Link Separators in Arrangements

A network of sensors is considered to be connected only if there is at least one path between each pair of nodes in the network. Connectivity depends primarily on the existence of paths. It is affected by changes in topology due to mobility, the failure of nodes, attacks and so on. The consequences of such occurrences include the loss of links, the isolation of nodes, the partitioning of the network, the upgrading of paths and re-routing. Connectivity can be modeled as a graph G (V, E) where V is the set of vertices (nodes) and E the set of edges (links). This graph is said to be k-connected if there are at least k disjoint paths between every pair of nodes u, v, V. Connectivity is a measure of fault tolerance or diversity of paths in the network. The need for 1-connectivity of the network graph is a fundamental condition of it being operational. Indeed, the connectivity of a network can be expressed as follows.

$$\mu(R) = \frac{N.\pi.R^2}{A},$$

where R is the radius of transmission, A the area and N the number of nodes in the area A.

Wireless sensor networks are commonly deployed in hostile environments and are susceptible to numerous faults in several layers of the system. Figure 1 depicts the source of these failures and demonstrates the potential for propagation to higher layers. The source of failures in this classification is divided in to four layers: node, network, sink and the base station. To address these problems it is useful to implement a system that allows monitoring of the network. At any moment such a system must be able to provide the operational status of different devices and to establish mechanisms that provide fault tolerance. By definition fault tolerance is a technique that has been proven to make systems capable of providing a good service, even in the presence of accidental phenomena such as disturbance of the environment (external faults), failure of hardware components (internal physical faults), or design faults, particularly software faults (bugs). Under the terms of dependability, faults are the causes of errors, mistakes are part of the abnormal state of the system and when errors are propagated to the system interface – i.e. when the service provided by the system is incorrect – this results in a failure. When mistakes are accidental and sufficiently rare, it is possible to tolerate them. This requires detecting errors before they occur, with error handling in case they can't be rectified. We must also make a diagnosis, in other words identify the fault, isolate faulty components, replace or repair and reset the system in case there is no alternative. In a wireless sensors network, fault tolerance is the ability to ensure the functionality of the network in the face of any interruption due to failures of sensor nodes.

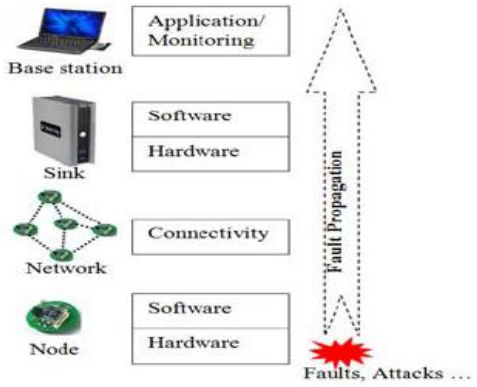

Figure 1: Fault tolerance and Propagation

3. Problem Statement and Solution

Consider a sensor network modelled as an undirected graph G = (V ,E), whose node set V represents the sensor nodes and the edge set E consists of pairs 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 Vcut ⊂ V such that the removal of the nodes in Vcut and the edges that is incident on Vcut 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 (i.e., no other connected sub graph G′ c of G contains Gc as its sub graph). 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.

The DSSD algorithm consists of two phases. One is a state update law, which a simple iterative procedure to compute the node potentials in the electrical network (Gelec,1) when s Ampere current is injected at the source node and extracted through the nodes Vfict, with all the nodes in Vfict grounded. The source strength s is a design parameter. The other phase of the algorithm consists of monitoring the state of a node, which is used to detect if a cut has occurred. We now describe the two phases below. Note that the separation into two phases is merely for conceptual clarity, they are carried out simultaneously at every node.

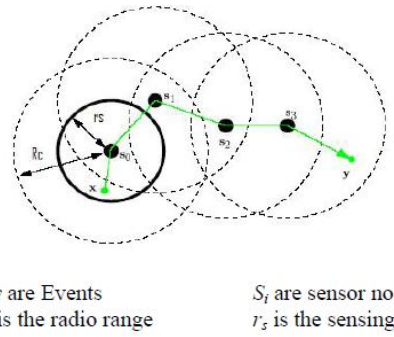

Figure 2: Connectivity and coverage in Wireless Sensor Networks

A. State update law

Let G(k) = (V (k),E (k)) denote 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. For ease of description, we index the source node as 1. Every node u maintains a scalar state xu(k) that is iteratively updated. At every iteration k, nodes broadcast their current states. Let Nu(k) = {v|(u,v) ∈ E (k)} denote the set of neighbours of u in the graph G(k). Every node in V except the source update its state as:

$$x_u(k+1) = \frac{1}{d_u(k)+1} \sum_{v \in \mathcal{N}_u(k)} x_v(k), \quad x_u(0) = 0,\ u \neq 1,$$

where du(k) := |Nu(k)| is the number of active neighbours of u at time k. If we count the fictitious node corresponding to u as one of u's neighbours whose state is held fixed at 0, then the above can be thought of as an average of the neighbours' states. The source node updates its state as:

$$x_1(k+1) = \frac{1}{d_1(k)+1} \left(\sum_{v \in \mathcal{N}_1(k)} x_v(k) + s \right) \quad x_1(0) = 0.$$

The description above assumes that all updates are done synchronously, or, in other words, every node shares the same iteration counter k. In practice, especially with wireless communication, an asynchronous update is preferable. To achieve this, every node keeps in its buffer a copy of the last received state of each of its neighbours. If in a particular iteration, a node does not receive messages from a neighbour during a time-out period, it updates its state using the last successfully received state from that neighbour. When a node fails, its neighbours will cease to receive messages from it permanently. When a node does not receive broadcasts from one of its neighbours for sufficiently long time, it removes that neighbour from its neighbour set. From then on, the node carries on the algorithm with the remaining neighbours.

B. 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 Gsource,G2,...,Gn, the component Gsource containing the source node. Since there is no source (and therefore no current injection) in each of the components G2,...,Gn 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 Gsource (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 ϱ that is provided a-priori. The parameters s and ϱ are design variables.

A major strength of the algorithm is that its convergence rate is independent of the number of nodes in the graph. This is particularly remarkable in view of the fact that the algorithm is purely distributed and employs only nearest neighbour communication. The convergence rate of distributed algorithms that use nearest neighbour communication, such as average consensus, rendezvous, decentralized formation control, etc. typically depend on the algebraic connectivity of the graph. The algebraic connectivity tends to decrease as the size of the graph increases, slowing down the convergence rate. In contrast, the DSSD algorithm's convergence rate is independent of the size of the network. The upshot of this property is that the delay between the occurrence of a cut and its detection can be bounded by a constant irrespective of the size of the network.

4. Robust and Energy Efficient Cut Detection

In this section, we present the theoretical foundations of cut detection and propose algorithms to enhance robustness and improve energy efficiency.

4.1 Preliminaries

We model our network as an undirected, connected graph $G = (V,E)$, where the set of vertices $V = \{v1, v2, ..., vm\}$ is the set of m nodes in the network and the set of edges $E = \{(vi, vj)|vi, vj \in V\}$ represents radio connectivity among nodes in the network. We denote by $Ni = \{vj \ |(vi, vj) \in E\}$ the set of *neighbors* of a node vi, and by $|Ni|$ the *degree* of node vi. Time is denoted as a discrete counter $k = 0, 1, 2, ...$ Each node vi maintains a positive real value $xi(k)$ which is called the state. The state is initialized to zero, i.e., $xi(0) = 0$ at time $k = 0$. One node in the network is designated as the *source node*. Although the *source node* may be selected arbitrarily, by convention we select the sink to be the source in WSN. For simplicity, we assume that $v1$ is the *source node*. At every iteration k, each node vi updates its state $xi(k)$ and broadcasts it. All nodes except the *source node* update their states using the following equation:

$$x_i(k+1) = \frac{1}{|N_i|+1} \sum_{j \in N_i(k)} x_j(k)$$

The *source node* $v1$ uses a slightly different state update equation:

$$x_1(k+1) = \frac{1}{|N_1|+1} \left(\sum_{j \in N_1(k)} x_j(k) + s \right),$$

where s, called the *source strength*, is a user specified scalar. Previously it was proved that the state of each node converges, after a number of iterations, to a positive value. We define a "cut" as a network partition, in which the graph G is separated into n disjoint connected components $Gsource, G2, ..., Gn$, where $Gsource = (Vsource, Esource)$ is a graph which contains the *source node*. When a "cut" occurs, the state of each node v $Vsource$ converges to 0. The convergence of a node's state is illustrated. Around iteration 40, the scalar state of nodes in the network converges. Shortly after iteration 60, a cut occurs in the network when the two nodes in the middle fail. After the cut, the state of a node on the right side rapidly decays to 0 while the state of a node on the left side converges to a new higher state. A critical observation is that the states of all nodes converge to new values, hence *all nodes* have the ability to detect a cut in the network. One troublesome aspect of cut detection using this distributed algorithm is that it is susceptible to attacks. A malicious node located in the disconnected part of the network can imitate a source node, and hence affect the state value that each node computes.

4.2 Robust Cut Detection Algorithm

Temporary variations of a node's state, often caused by packet loss, can be tolerated by a system implementing cut detection as described above. The states of nodes in the network will eventually converge. However, this is not true when a non-source node continuously injects a constant state to the system. This malicious source node is formally defined as:

Definition: A node $v_i \in G$ is a malicious node M_i if it acts as a source node in the network, i.e., it

updates its state according to equation 2 with an arbitrary strength s_, as given by

$$x_i(k+1) = \frac{1}{|N_i|+1}\left(\sum_{j \in N_i(k)} x_j(k) + s'\right)$$

Algorithm 1. Compute Distribution

Input: S, p
Output: μ, σ
  $S_i^N \leftarrow \emptyset$, $min\_dist \leftarrow 0$, $min\_idx \leftarrow 0$
  for $i = 0$ to $|S|$ **do**
    for $k = 0$ to p **do**
      for $j = 0$ to $|S|$ **do**
        if $i \neq j$ and $s_j \neq -1$ **then**
          $dist \leftarrow |s_i - s_j|$
          if $dist < min\_dist$ **then**
            $min\_dist \leftarrow dist$
            $min\_idx \leftarrow j$
          end if
        end if
      end for
      $s_{min\_idx} \leftarrow -1$
      $s_i^N \leftarrow s_i^N + min\_dist$
    end for
  end for
  $\mu \leftarrow \frac{\sum_i s_i^N}{|S|}$
  $\sigma \leftarrow \sqrt{\frac{\sum_i (s_i^N - \mu)^2}{|S|-1}}$

Algorithm 2. Outlier Detection

Input: s_i^N
Output: *True, or False*
  if $\frac{|s_i^N - \mu|}{\sigma} > t\_table[|S|]$ **then**
    return *True*
  end if
  return *False*

5. Conclusion

A wireless sensor network can get separated into multiple connected components due to the failure of some of its nodes, which is called a "cut". 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 every node to detect when the connectivity to a specially designated node has been lost, and 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.

References

[1]. Prabir Barooah, Harshavardhan Chenji, Radu Stoleru, and Tam´as Kalm´ar-Nagy – "Cut Detection in Wireless Sensor Networks", IEEE Transaction on Parallel and Distributed Systems,2012.

[2]. Nisheeth Shrivastava Subhash Suri, Csaba D. T´oth – "Detecting Cuts in Sensor Networks".

[3]. Benahmed Khelifa, H. Haffaf , Merabti Madjid3, and David Llewellyn-Jones – "Monitoring Connectivity in Wireless Sensor Networks", International Journal of Future Generation Communication and Networking Vol. 2, No. 2, June, 2009.

[4]. Jon Kleinberg Mark Sandler Aleksandrs Slivkins – "Network Failure Detection and Graph Connectivity", June, 2003 Minor revision: July 2007.

[5]. Myounggyu Won, Stephen M. George, and Radu Stoleru – "RE2-CD: Robust and Energy Efficient Cut Detection in Wireless Sensor Networks", B. Liu et al. (Eds.): WASA 2009, LNCS 5682, pp. 80–93, 2009. Springer-Verlag Berlin Heidelberg 2009.

[6]. Prabir Barooah – "Distributed Cut Detection in Sensor Networks", Proceedings of the 47th IEEE Conference on Decision and Control Cancun, Mexico, Dec. 9-11, 2008.