TEACHING WIRELESS SENSOR NETWORKS BY INCORPORATING REORGANIZATION ALGORITHMS INTO THE LABS

T. Andrew Yang
yang@uhcl.edu

Dinesh Reddy Gudibandi
dinesh_gudibandi@yahoo.com

Division of Computing and Mathematics
University of Houston – Clear Lake (UHCL), Houston, Texas

Abstract

Wireless sensor networks (WSNs) have emerged in recent years as a topic attracting significant interest from both academia and industry. By completing a WSN lab, students can learn useful concepts and skills, including network setup, protocols, data collection/processing, sensor programming, developing and configuring WSN, etc. To effectively teach wireless sensor networks, having students engaging in hands-on projects is as important as teaching them the theories. Our goal is to build upon the source codes of an existing WSN method and design a set of simulation and emulation labs, each incorporating a case study involving tasks such as reorganization, secure routing, object tracking, Denial of Service attacks and counter-measures, etc. Instead of creating a WSN from the scratch, students learn about WSN via tailoring a method to incorporate interesting features into the existing source codes. In this paper, we propose to use WSN reorganization as a case study and incorporate it into designing WSN simulation labs. Starting with a given set of WSN papers, source codes, and reorganizaiton algorithms, students are asked to incorporate some of the algorithms into the existing WSN method. Such projects enable the student to gain in-depth understanding of how a WSN operates, and how the source codes of an existing WSN method may be modified to enable reorganization of sensor nodes. A sample lab incorporating WSN reorganization is presented as a demonstration of this approach. Interested instructors may tailor the lab to accommodate other case studies for their classes.

1. Introduction

A wireless sensor network (WSN) is comprised of a set of sensor nodes coordinated to fulfill some specific actions. Deploying a WSN requires iteratively programming a set of nodes, positioning them in an area large enough to produce an interesting radio topology, and using them to extract debugging and performance data [6]. As we have learned from experiences, it is often challenging for a student with no prior experiences in WSN to deploy or test such a network. Our goal is to develop a set of WSN labs that could be easily adopted by the instructor, and could be used to help the students get started in testing and/or deploying wireless sensor networks.

Although both simulation and emulation may be employed when developing the labs, we focus on simulation-based labs in this paper. Equipped with the source codes of an existing WSN method, Optimized Communication and Organization (OCO) [5], our approach is to design a set of labs, each incorporating a case study such as reorganization, secure routing, Denial of Service attacks and counter-measures, etc., in WSN. Instead of creating a brand new WSN method from the scratch, students learn about WSN via tailoring an existing method to incorporate interesting features into the existing source codes. The labs may be used in a standalone WSN class, or incorporated into other related classes such as Network Protocols, Advanced Operating Systems, Network Security, and the Capstone Projects course.

In this paper, we focus on incorporating WSN reorganizaiton into the design of simulation-based labs. Students are given the source codes of an existing WSN method and a survey of WSN reorganizaiton algorithms. To complete their lab projects, the students are required to incorporate some of the reorganizaiton algorithms into the existing codes (for intermediate level projects) or to develop and implement their own reorganizaiton algorithms (for advanced projects in courses such as a graduate WSN class). The projects enable the student to gain an in-depth understanding of how a wireless sensor network operates, and how it may be reorganized when a need arises.

Depending on the instructor's interests, the same approach may be used to handle other WSN case studies.

In the rest of the paper, we first survey related work in reorganizing WSNs. In section 3 we present a case study, by first briefly introducing the OCO method, the requirements of WSN reorganizaiton in various scenarios, an example reorganizaiton algorithm in OCO, and finally a sample lab design incorporating the case study. Section 4 concludes the paper.

2. Reorganization of Wireless Sensor Networks

Once deployed, a WSN typically operates unattended. Furthermore, a WSN may be used in hostile or inaccessible environments, such as in some military or rescue operations. The sensors may even be deployed by using airplanes/artillery in a hard-to-reach territory, making it virtually impossible to repair a node when it is damaged or run out of energy.

An important requirement of designing a WSN is to ensure continued operation even when some of the nodes become unavailable. When a WSN is first deployed, most of the sensor nodes are charged with full energy. Because data traffic are typically routed from the sensors to the sink (aka. the *base* station), more energy is consumed by the nodes near the sink, making that neighborhood more susceptible to energy depletion and failure. A reorganization algorithm can ensure the WSN could recover from damage, loss, or degradation of some of its nodes.

The topology of a network can be classified as either flat or hierarchical. In a flat network, all the sensors have similar roles [3]. Each node typically plays the same role and collaborate together to perform the sensing task [1]. In a hierarchical sensor network, nodes may be assigned different roles and, depending on their respective levels in the hierarchy, the processing at a particular node may differ. LEACH based methods [2] form hierarchical sensor networks.

Reorganization in a flat network is generally simpler, because all nodes are at the same level and a failing node may be easily replaced by another node. A WSN with a hierarchical structure, however, is more difficult to reorganize in the event of node failure. Depending on the failing node's role, the reorganization requires not only choosing the proper node to replace the failing node, but also reorganizing the network topology. Efficient Data GathEring (EDGE) [4] is a tree-based protocol. The tree created by EDGE will be reconstructed upon node failures or adding of new nodes. When a node is aware of the absence of its parent, it immediately switches to a new parent by choosing the most appropriate parent from its PC (*parental candidate*) table[1].

3. The Case Study: WSN Reorganization

In this section we discuss the OCO method (Optimized Communication and Organization) [5], its reorganization algorithms and their respective requirements, and a sample lab project that incorporates WSN reorganization as a case study.

OCO is a WSN method that ensures maximum accuracy of target tracking, efficient energy dissipation, and low computation overhead [5]. An OCO-based WSN is built with self-organizing capabilities, meaning that when some of the sensor nodes become unavailable, the WSN may reconfigure itself to continue its mission. As shown in Figure 1, the OCO method includes four phases: *position collection*, *processing*, *tracking*, and *maintenance*. In the *position collection* phase, the base station collects positions of all reachable nodes in the network. Figure 2 shows snapshots of the position collection phase during simulation using OMNET++[2]. The red small

[1] Parental candidate (PC) table is a table that maintains the pairs of candidate IDs (i.e., those that can be a potential parent) and metrics.

[2] The OCO codes were developed in C++ for the OMNeT++ network simulation environment. As stated at the omnetpp.org website, OMNeT++ is a public-source, component-based, modular and open-architecture simulation environment with strong GUI support and an embeddable simulation kernel. Its primary application area is the simulation of communication networks, and because of its generic and flexible architecture, it has been successfully used in other areas like the simulation of IT systems, queueing networks, hardware architectures, and business processes as well.

circles in Figure 2 are nodes already being recognized by the base (the bigger circle). As shown in the diagrams (from left to right), more nodes are recognized when the process continues.

The *processing* phase identifies redundant nodes, detects border nodes, finds the shortest path from each node to the base, and assigns various roles to the nodes. The process of cleaning up redundant nodes results in a certain percentage of the nodes being included in the network (the *active* nodes), while the others (the *redundant* nodes) become inactive to conserve energy. Figure 3 include two images illustrating the process of cleaning up redundant nodes. Image a is the initial sensor network, showing numerous nodes in the sensing area; image b is the sensor network after redundant nodes have been removed.

In the *tracking* phase, the sensors all work together to detect and track intruders. The *maintenance* phase involves re-organizing the network when, for example, a change in the topology of the network occurs, or some of the sensor nodes become dead. Some of the *redundant nodes* previously labeled as inactive may become active during the reorganization process.

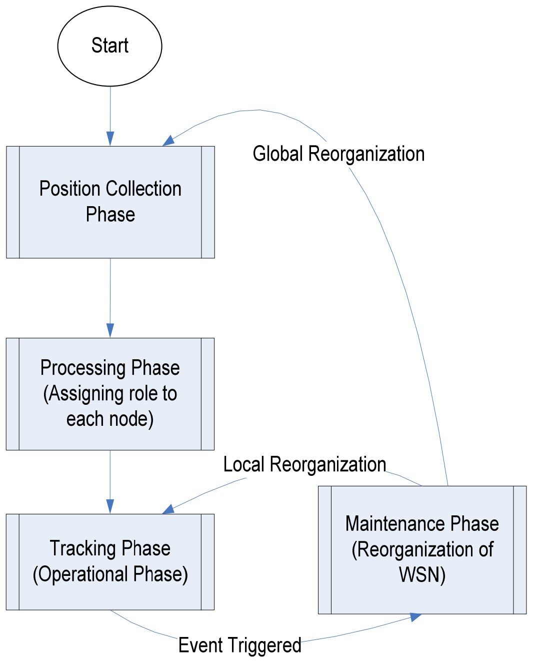

Start
Position Collection Phase
Global Reorganization
Processing Phase (Assigning role to each node)
Local Reorganization
Tracking Phase (Operational Phase)
Maintenance Phase (Reorganization of WSN)
Event Triggered

Figure 1: OCO phases and their interactions

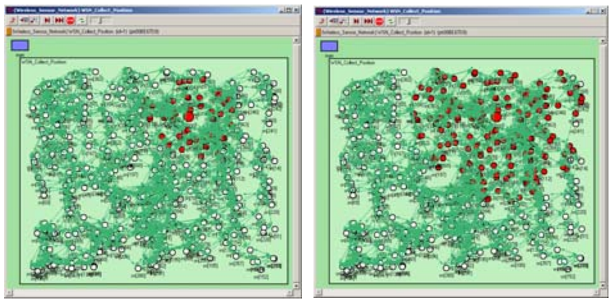
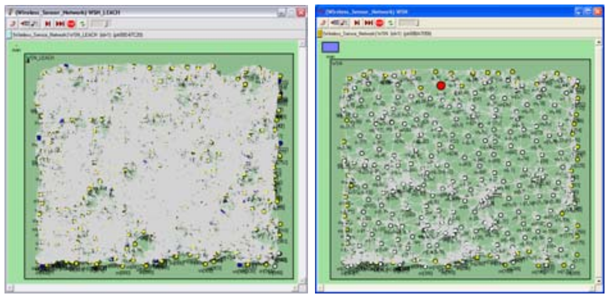


Figure 2: The *position collection* phase



(a) before (b) after
Figure 3: Removing redundant nodes

3.1. Reorganization algorithms in OCO and the Requirements

There are different types of reorganization algorithms in OCO, each of which is to be invoked to reorganize sensor nodes under a particular circumstance. As outlined in Table 1, when a given condition is true, certain reorganization algorithm is invoked; based on the underlying assumptions (if any), certain outcome is anticipated from running a given algorithm.

Table 1. Types of reorganization algorithms and the respective scenarios

Condition	Assumptions	Outcome
Algorithm 1: Border node reorganization		
When a border node is dead or about to die.	*A1:* The Base station has the map of all nodes (active or redundant) and the paths of all active nodes to the Base station. *A2:* Because the random deployment of the sensor nodes, for a given active node, there is a high probability that several redundant nodes may be lying around the dead	One (or more) redundant node close to the dead node (*B*) takes place of *B*, by filling up the gap

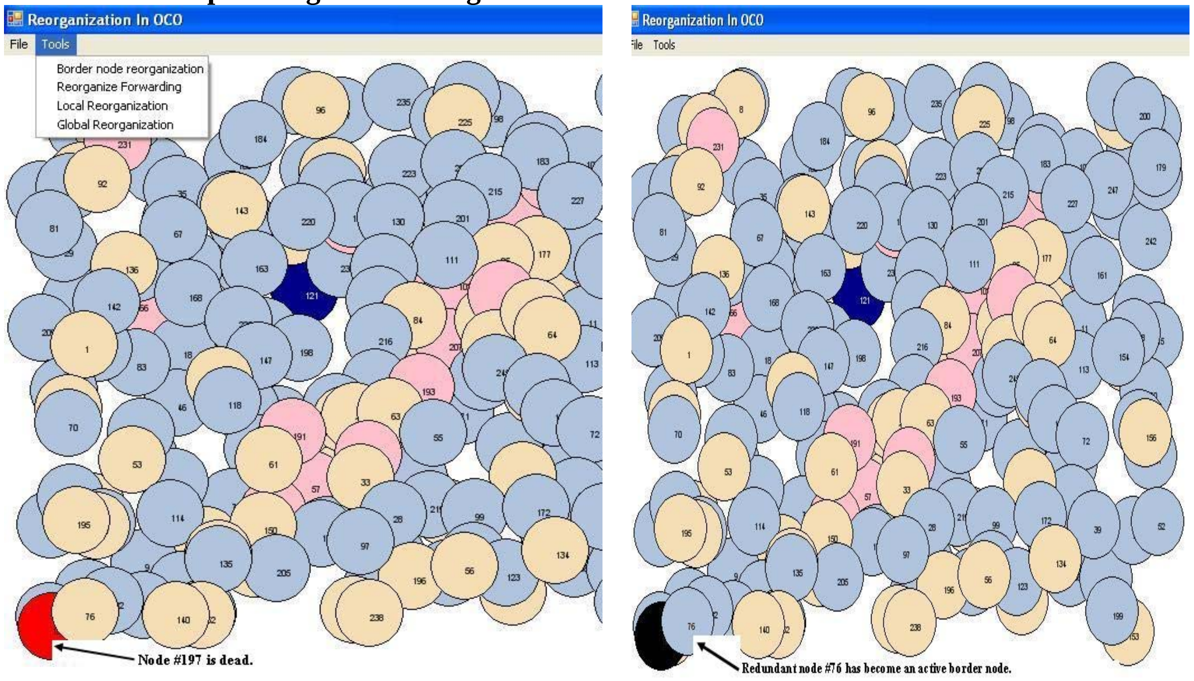
	node.	left by *B*.

Algorithm 2: Forwarding node reorganization		
When a single forwarding node is dead or about to die.	*A1* *A2*	Case A: One (or more) neighboring node (N) near the dead node (D) is chosen to be the new parent of D's children. Case B (When Case A does not qualify): One (or more) redundant node (R) near D is made active and become the new parent of D's children.

Algorithm 3: Local reorganization		
To reorganize a region of the network when several nodes in a sub-tree are reported to be dead or about to die.	A1 A2 A3: All the border nodes are sequentially ordered, meaning the Base is able to find out the two border nodes on the two sides of a given border node. A4: For a given sub-tree, the Base can easily figure out the set of border nodes (the leaves) of that sub-tree.	Several redundant nodes (*R*) in the region are made active and become the new parents of the children of the dead nodes (*D*), so that the whole area affected by *D* is covered by the newly constructed sub-trees.

Algorithm 4: Global reorganization	
If there are a considerable number of dead sensor nodes in a WSN, the entire network may need to be reorganized.	The WSN is reorganized globally, repeating the first 3 steps of OCO.

3.2. An example reorganization algorithm in OCO



(a) before **(b) after**

Figure 4. Border node reorganization

(invoked by the death of border node #197, which is replaced by redundant node #76)

Due to limitation of space, we cannot include in this paper all the reorganization algorithms as listed in Table 1. As an example, a simple reorganization algorithm is presented below; it is used to reorganize an OCO-based WSN when a single border node is dead.

<u>Border node reorganization algorithm</u>

(1) Find one or more redundant nodes **r** around dead node **n** such that the new sensing range of **r** can cover the sensing range of the dead node **n**.

(2) Put the new nodes to list **l.**
(3) For each of the nodes in list **l**, form paths to the Base.
(4) The Base then informs each of the nodes affected by the reorganization about its new parent and children nodes.

Figure 4 illustrates the impact of running that algorithm when a border node is dead. As shown in Figure 4a, the border node reorganization algorithm is applied when it is detected that border node #197 is dead. The reorganizaiton algorithm selects a redundant node (#76) near the dead node, and uses it to replace the dead border node. As a result, the dead node is marked as *dead* (as shown in Figure 4b) and removed from the network. The broken border line caused by node #197 being dead is successfully 'patched' by adding node #76 as a new border node.

3.3. Incorporating the case study into the labs

In this subsection, we present an example simulation lab that incorporates WSN reorganization. As stated above, reorganization is chosen as a case study of wireless sensor networks. Depending on the instructor's interests, alternative case studies involving, for example, routing, object surveillance or tracking, Denial of Service attacks, etc., may be used to design the labs.

Lab Title: Testing or developing reorganization algorithms to reorganize sensor nodes in a wireless sensor network.

Target Classes: The lab may be used in upper level undergraduate classes like Network Protocols, Wireless Sensor Networks, etc., or in graduate classes involving Wireless Sensor Networks.

Learning objectives: Students will learn the following after having successfully completed the lab.

- How a wireless sensor network operates in general.
- How sensor nodes are organized into a wireless sensor network.
- How different *roles* are assigned to the various nodes in the same wireless sensor network.
- How redundant nodes close to the dead node attain connections to the children of the dead node.
- How different reorganization algorithms are invoked to handle different kinds of dead nodes.
- How different reorganization algorithms may be developed to handle different conditions.

Tools Utilized: Visual Studio, OMNeT++ network simulator, and the OCO codes.

Requirements: Each student is given the OCO source codes, the relevant papers, the reorganization algorithms, and the related document explaining under what condition a particular reorganization algorithm would be invoked. Depending on the level of difficulty of the project assigned to a student, the student is required to complete one of the following tasks:

a) *Intermediate* level (e.g., upper-level undergraduate students) - Students are required to incorporate the given reorganizaiton algorithms into the existing OCO codes. Anticipated Outcome – Source program(s) implementing the algorithms, screen output, and a demo of the implemented programs executing in the OMNET++ simulation environment.
b) *Advanced* level (e.g., advanced undergraduate or graduate students) - Students are required to test the reorganization algorithms, and to develop their own reorganization algorithms and integrate them into the OCO codes. Anticipated Outcome – Design of the new set of reorganization algorithms, the source program(s) implementing the algorithms, screen output, and a demo of the implemented programs executing in OMNET++.

Problem classification: The application can be classified as a study and programming project, because it involves studying papers and source codes, and developing/implementing algorithms.

How it may be implemented in the lab: Students will use *Visual Studio* to revise the existing OCO codes, and incorporate the reorganizaiton algorithm(s) into OCO.

Level of difficulty: The level of difficulty of the project is intermediate (when implementing a given algorithm) or advanced (when designing and implementing a new algorithm).

Grading criteria and methods: Grades mainly depend on the successful development and implementation of the reorganization algorithm(s), as well as documentations, presentations, etc.

Students' experiences: As reported by students, working on the labs allows the students to learn the fundamentals of WSN with respect to how a WSN works, how the nodes are classified as a border node, forwarding node, etc., and how sensed data are aggregated and sent to the controlling station. They also have learned the importance of reorganization in the WSN, and how to use the OMNeT++ network simulator. Besides, the students have reported that their overall programming skills have improved.

4. Conclusion and Future Work

By focusing on a case study when designing wireless sensor networks lab projects, we present a feasible approach of including hands-on simulation projects into a class, providing students the opportunities to develop in-depth understanding of how a wireless sensor network operates, and how to modify an existing WSN method to incorporate new algorithms. Instead of requiring the students to develop a wireless sensor network from the scratch, our approach provides the students the source codes of an existing algorithm, allowing the student to develop in-depth understanding of WSN by engaging in adding additional features into the existing WSN method. For beginning or intermediate students, the reorganization algorithms may be given to them; the anticipated outcome is the successful implementation and integration of the given algorithm(s) into an existing system. For advanced students, the project may involve having the students develop their own algorithms; the anticipated outcome is the design and implementation of new algorithm(s), and the integration of the algorithm into an existing system. Both options allow the students ample opportunities to learn about the fundamentals of networking, programming, design and analysis, and advanced knowledge of wireless sensor networks.

This project may be extended to cover other case studies, such as network routing, attacks and counter-measures, etc. In addition, the case-study based approach may be adapted to emulation based labs, in which actual devices are used to construct a sensor network.

Acknowledgement

The authors are partially supported by the UHCL Faculty Research Support Fund, the Texas Advanced Research Program, and the National Science Foundation (DUE- 0633469).

References

[1] Al-Karaki, J. N., and A. E. Kamal. “Routing Techniques in Wireless Sensor Networks: A Survey”. *IEEE Wireless Communications*, Vol. 11, Issue 6, pp. 6-28, Dec. 2004.

[2] Heinzelman, Wendi Rabiner, Anantha Chandrakasan, and Hari Balakrishnan. “Energy-Efficient Communication Protocol for Wireless Microsensor Networks”. *Proceedings of the Hawaii Int. Conf. on System Sciences*, 2000.

[3] Ma, Y., S. Dalal, M. Alwan, and J. Aylor. “ROP: A Resource Oriented Protocol for Heterogeneous Sensor Networks”. *University of Virginia, Virginia Tech Symposium on Wireless Personal Communications*, 2003.

[4] Thepvilojanapong, Niwat, Yoshito Tobe, and Kaoru Sezaki. “On the Construction of Efficient Data Gathering Tree in Wireless Sensor Networks”. *IEEE Communications*, 2005.

[5] Tran, Sam Phu Manh, and T. Andrew Yang. “OCO: Optimized Communication & Organization for Target Tracking in Wireless Sensor Networks”. *Proceedings of the IEEE Int. Conf. on Sensor Networks, Ubiquitous, and Trustworthy Computing*. 2006.

[6] Werner-Allen, Geoffrey, Patrick Swieskowski, and Matt Welsh. “MoteLab: A Wireless Sensor Network Testbed”. In *Proceedings of the Fourth International Conference on Information Processing in Sensor Networks (IPSN'05)*, Special Track on Platform Tools and Design Methods for Network Embedded Sensors (SPOTS), April 2005.