Evaluations of Target Tracking in Wireless Sensor Networks

Sam Phu Manh Tran T. Andrew Yang
University of Houston – Clear Lake
2700 Bay Area Blvd., Houston, Texas 77058
(281) 283-3835
tpmsam@yahoo.com yang@uhcl.edu

ABSTRACT

Target tracking is one of the most important applications of wireless sensor networks. Optimized computation and energy dissipation are critical requirements to maximize the lifetime of the sensor network. There exists a demand for self-organizing and routing capabilities in the sensor network. Existing methods attempting to achieve these requirements, such as the LEACH-based algorithms, however, suffer either redundancy in data and sensor node deployment, or complex computation incurred in the sensor nodes. Those drawbacks result in energy use inefficiency and/or complex computation overhead. *OCO*, or *Optimized Communication and Organization*, is an algorithm that ensures maximum accuracy of target tracking, efficient energy dissipation, and low computation overhead on the sensor nodes. Simulation evaluations of OCO are compared with other two methods under various scenarios.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: *Wireless Communication, Sensor Networks*

General Terms

Measurement, Performance, Design, Experimentation

Keywords

Sensor Network, Object Tracking

1. INTRODUCTION

Wireless sensor networks have significant impact upon the efficiency of military and civil applications such as environment monitoring, target surveillance, industrial process observation, tactical systems, etc. In these scenarios, *target tracking* is one of the most important applications of wireless sensor networks.

The simplest way for target surveillance is to turn on the sensor module of all the nodes in the network and have each node communicate directly with the base, the so-called *Direct Communication* (DC) method.

The DC method gives the best accuracy in tracking objects. However, it is unrealistic because the base has a limited number of channels. In addition, the node's communication distance is limited, so this method is not applicable to a large area.

Another issue of efficient target tracking in sensor networks is redundancy. Because nodes are typically deployed at random locations (for example, thrown by an airplane), it leads to overlapping among sensing areas of the nodes.

Existing computer network protocols may not be applicable to sensor networks, because sensor nodes are constrained in energy supply, performance, and bandwidth. Existing methods attempting to alleviate these constraints, such as the LEACH-based algorithms [2], however, either suffer redundancy in data and sensor node deployment, or require complex computation in the sensor nodes. Those drawbacks result in energy use inefficiency and/or complex computation overhead. Therefore, there exists a demand for self-organizing and routing capabilities in the sensor network, in order to achieve optimized computation and energy dissipation, and to maximize the lifetime of the sensor network.

In this paper, we present the results of simulation-based evaluations of three algorithms for sensor networks. The first is the Direct Communication (DC) method [1]; the second is the well-known LEACH [2], while the third is a method we devise, OCO (or Optimized Communication and Organization). OCO ensures not only maximum accuracy of target tracking, but also efficient energy dissipation and low computation overhead.

In the next section, the three methods are overviewed[1]. In section 3, we discuss the simulation models, tools, and metrics used in evaluating the three methods. Results of the evaluations under various scenarios are discussed in section 4.

2. THE EVALUATED METHODS

In DC, sensor modules of all nodes are ON and nodes send information about the intruders to the base directly.

There are 2 phases in LEACH: set-up phase and steady-phase. In the *set-up phase*, sensors may elect randomly among themselves local cluster heads. After the set of cluster heads are selected, each of the cluster heads advertises to all sensor nodes in its communication area that it is the new cluster head. Once a node receives such advertisements, it decides to which cluster head it would belong. Finally, the cluster head assigns the time slot on which the sensor nodes can send data to them. In the *steady-phase*, sensors begin to sense and transmit data to the cluster heads. After a certain period of time spent in the *steady* state, the network is refreshed by entering the *set-up* phase again.

SIGCSE'06, March 1–5, 2006, Houston, Texas, USA.

[1] Due to space limitation, detailed discussion of the OCO method is not possible in this paper. Interested readers may contact the authors to obtain the detailed algorithms for the various phases.

OCO has 4 phases: position-collecting, processing, tracking, and maintenance.

The *position-collecting phase* involves the base's collecting positions of all the nodes in the network. The *processing phase* involves the base's cleaning up the redundant nodes, detecting the border nodes, and routing. The *tracking phase* detects all objects coming from outside the perimeter of the sensor network. Normally, only the sensor modules of the border nodes are ON. When a border node detects an object, it periodically sends its position information to the base by using its father's ID. When it has lost the object, it sends a message to turn on the sensor modules of all its neighbors[2]. If a neighbor detects the object, it will continue sending its position to the base and, right after it has lost the object, it turns the sensor modules of all its neighbors to ON, and so on. If activated neighbors detect nothing, they automatically turn the sensor module OFF after a short interval. This way, the objects are tracked as long as it remains within the network perimeter. The *maintenance phase* is started when the network has a dead node (for example, the node has run out its energy). In this case, the base just deletes the dead node from the list, and then re-organizes the network by starting the 4-step procedure over.

3. SIMULATION AND EVALUATIONS

3.1. Models and Tools

The simulation models that we have built to test the performance of the three methods are based on [3], and consist of three sub-modules: a sensor-node model, a sensor-network model, and an intruder-object model.

The sensor-node model is used to simulate a sensor node. It includes three layers: application, MAC, and physical layer. It contains several modules, including the coordinator module, the radio module, the sensor module, and the energy module, as illustrated in Figure 1. Physical layer represents the interfaces with other nodes. The MAC layer represents pre-processing packet layers. In the simulations, the performance of the MAC[3] layer is not considered.

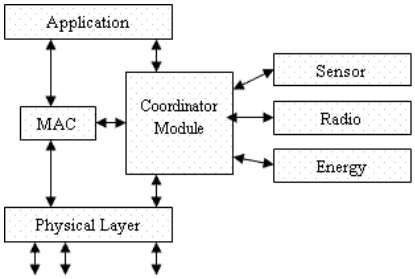


Figure 1: A sensor-node model

The sensor-network model contains a set of sensor nodes, and is used to represent the sensor network. Two nodes can communicate with each other if the distance between them is smaller than their communication radius.

The intruder-object model represents the intruding object. Although being similar to the sensor node, an object has only two layers: the application layer on top of the physical layer.

[2] It is assumed that the delay time to activate the neighbors is smaller than the value of the sensing radius divided by the speed of the intruding object.

[3] The MAC layer module is actually disabled to speed up the simulation.

The tools used to implement the models include C# (for processing programs), OMNeT++ (for simulation), and MatLab (for analyzing and evaluating simulation results).

3.2. The Metrics

Four metrics are used in evaluating the methods:

- *Energy consumption* measures the total energy consumed by the sensor nodes after the simulation is started.
- *Accuracy* is the number of detected positions of the intruding object(s) in a given method, compared to the number of detected positions in the DC method, which is used as the base of comparisons [4].
- *Cost per detected position* is the ratio between *energy consumption* and the number of detected positions.
- *Time before the first dead node* is the time when the first node of the network runs out of energy [2].

3.3. Simulation Environment

The simulation environment is built as an area of 640x540. The number of nodes in the network is 200, 250, 300, 350, 400, 450, 500,550, 600, 650,700, 750, 800, 850, 900, 950, and 1000 with 2J (Joule[4]) of energy for each node. In each situation, nodes and the base station are placed randomly. The sensing radius and the communication radius of the nodes are set as 30 and 60 respectively. Table 1 is a summary of the respective consumed energy, with respect to various tasks.

Create/Receive a data message Create/Receive a signal message	100 µJ 3 µJ
Send a data message (d<= 60m) Send a signal message (d<=60m)	820 µJ 26 µJ
Send a message (d > 60m)	100 µJ + 0.1*d^2
Sensor board (full operation)	66 µJ/s
Radio board (idle/receive mode)	100 µJ/s

Table 1: Energy consumption for various tasks

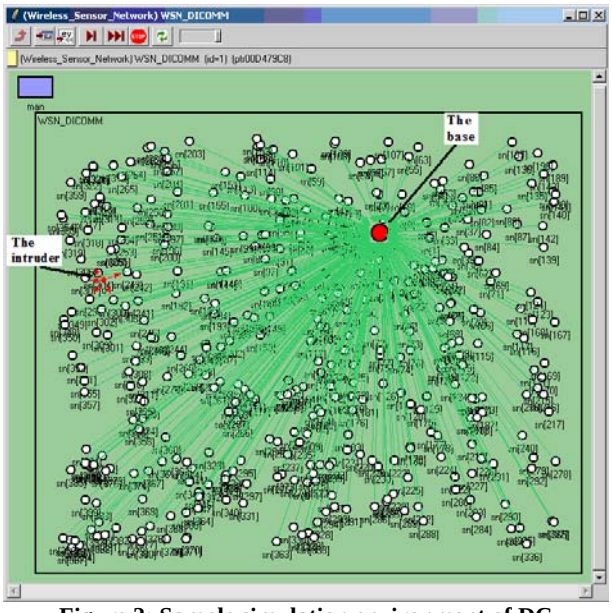


Figure 2: Sample simulation environment of DC

[4] A *Joule (J)* is the unit for measuring quantity of energy.
1 Watt = 1 Joule/second

Figures 2, 3, and 4 are the simulation screen shots of the sensor networks constructed respectively by the three methods. The big red circle in the figures represents the *base*, and the red dot with arrows pointing out represents the *intruder object*. In Figure 3 (LEACH), the blue nodes are the cluster heads. In Figure 4 (OCO), the yellow nodes represent the border nodes, and the orange nodes the activated nodes.

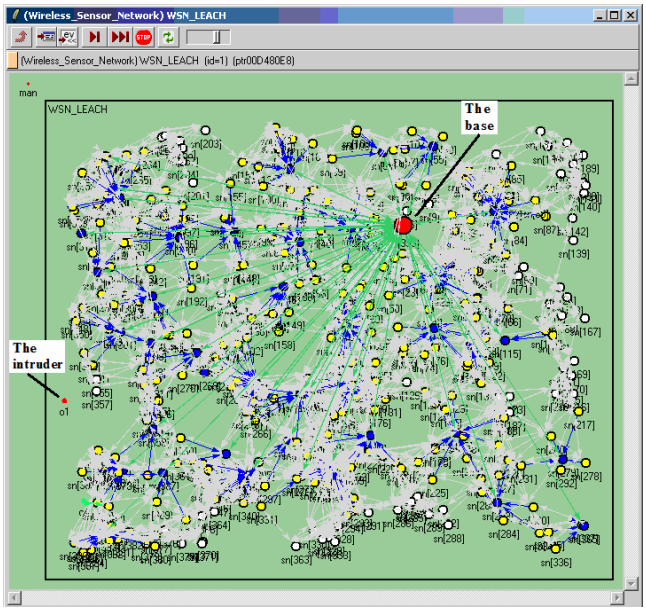


Figure 3: Sample simulation environment of LEACH

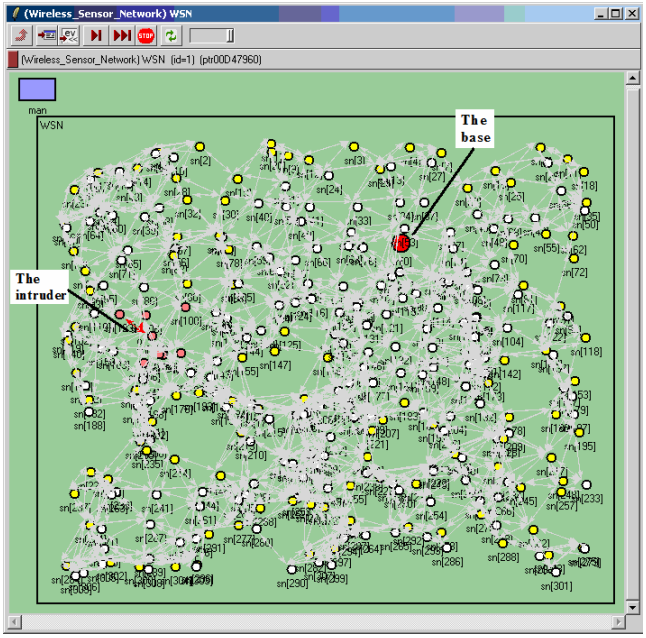


Figure 4: Sample simulation environment of OCO

4. RESULTS

Results from two different cases are presented in this section. In the base case, the three methods are evaluated in environments with no intruder object present; in the second case, one intruder is present in the simulation environments.

4.1. The Base Cases: no intruder objects

In the base case, two metrics, *Energy consumption* and *Time before the first dead node*, are measured. The results of running the three methods in 9000 seconds of simulations are shown in Figures 5 and 6. As shown in figure 5, when the number of nodes increases, the energy dissipation of OCO goes to a constant, delivering much better result than the other methods. The only exception is when the number of node is 200, which is the only case when OCO's value is higher. The reason is that, when the number of node increases, the size of the border in OCO decreases (less gaps between border nodes and straighter border line). The minimum is reached at 950.

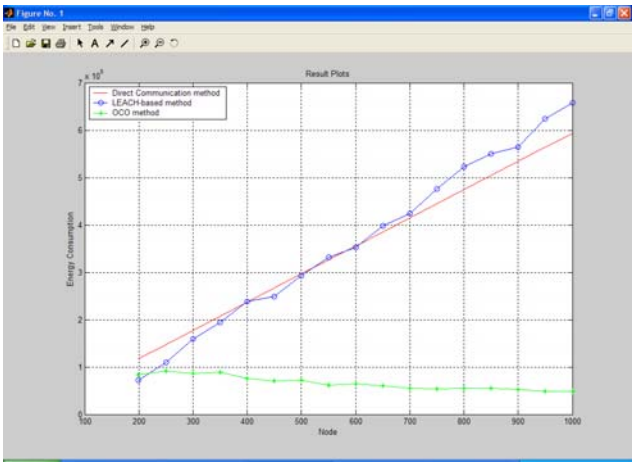


Figure 5: *Energy Consumption* (no intruder)

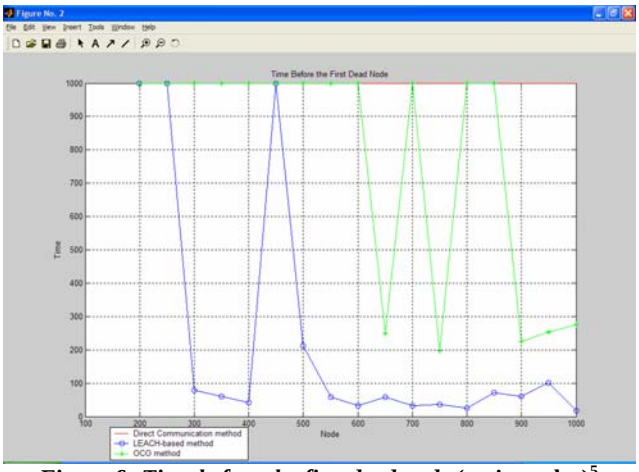


Figure 6: *Time before the first dead node* (no intruder)[5]

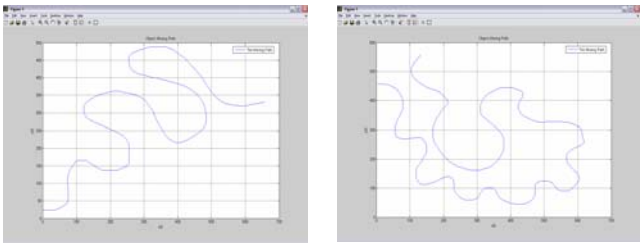
Time before the first dead node, as illustrated in Figure 6, shows that OCO lasts longer than LEACH in all cases. Although DC shows superior performance in this regard, it is not a practical method, and is thus used mainly as a base of comparison.

4.2. The Tracking Cases: one intruder object

In this section, the simulation results are divided into two cases. Each of the cases represents the collected metrics given a particular path of the moving object (Figure 7), with the moving speed being 10 points/s. The paths are created by drawing images,

[5] Value of 1,000 means there is no dead node in the network during the 9000 seconds of simulation.

which is then read by a Mathlab script to generate text files of all points belonging to the paths. In each of the cases, four figures are presented to show the results with respect to the four metrics.



Case 1: Diagonal zigzag path Case 2: Along-the-border path

Figure 7: The paths for different cases

4.2.1. Results of Case 1

The path used in this case represents the scenario where the moving object moves along a zigzag course diagonally (lower left to upper right) across the sensor network (Figure 7, case 1).

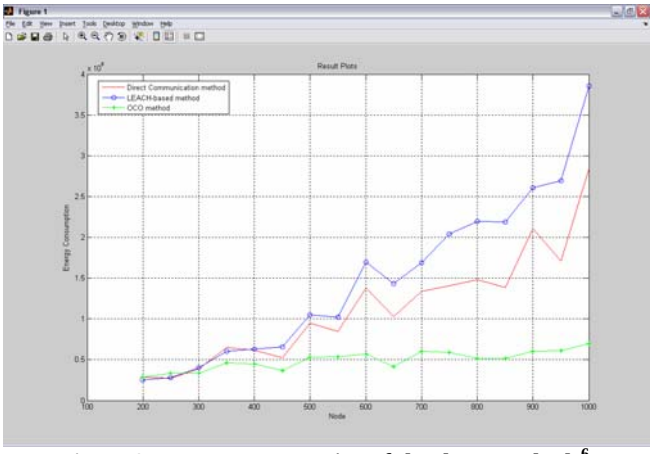


Figure 8: *Energy consumption* of the three methods[6]

In Figure 8, OCO delivers more stable results than the others in all cases except when the number of nodes is smaller than 300. Our explanation is that, in those cases, there are very few nodes in the area, so most of the nodes are ON. It means OCO is close to DC or LEACH in terms of the number of ON nodes; however, OCO has to spend energy in colleting node positions. That is why OCO would consume more energy than the others when the number of nodes is less than 300.

As shown in Figure 9, the *accuracy* of OCO is compatible to DC in most of the cases. LEACH, although showing very good results when the number of nodes is higher than 700, exhibits less stable results when the number of nodes is lower than 700.

Figure 10 shows the cost associated with each of the detected object positions. OCO's cost is the lowest in all the cases when the number of nodes is greater than 300. The cost of LEACH in general increases relative to the number of nodes.

As shown in Figure 11, the *time before the first dead node* appears to be "fluctuating" across the cases, because, depending on the relative distance between the intruding object and the base, one of the nodes may run out of energy early. Still, as shown in Figure 11, an OCO network lasts longer than LEACH in all cases.

[6] The "spikes" exhibited by DC and LEACH at 600 and 1,000 were caused by the fact that the base station in those cases were far from the moving paths, resulting in more energy consumption.

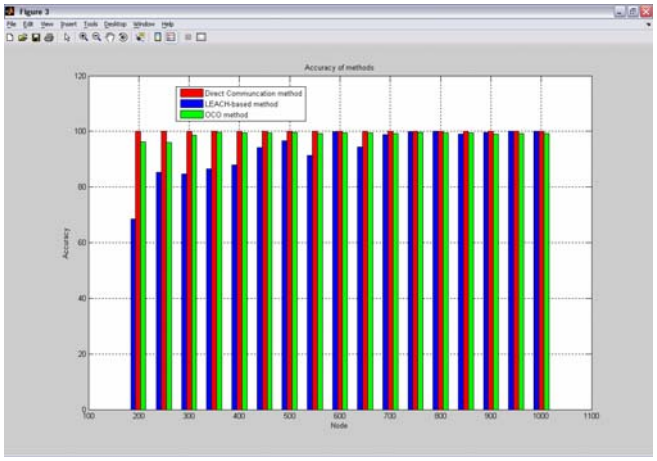


Figure 9: *Accuracy* of the three methods

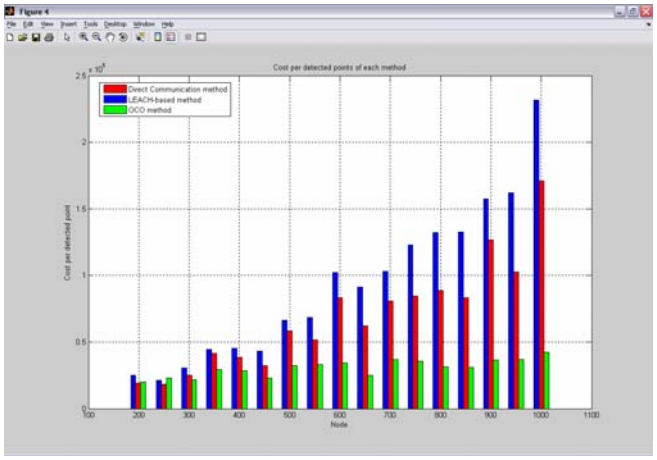


Figure 10: *Cost per detected point* of the three methods

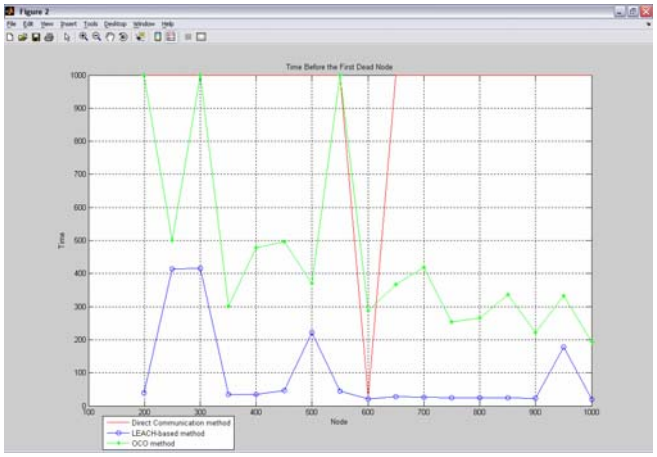


Figure 11: *Time before the first dead node* of the three methods

4.2.2. Results of Case 2

The path used in this case represents the scenario where the moving object moves approximately along the border of the sensor network, starting at the upper left corner toward the lower right corner, and then coming back to the upper left corner (Figure 7, case 2). Results of case 2 are compatible with case 1. As shown in Figure 12, *energy consumption* of OCO is smaller than the others when the number of node is greater than 300.

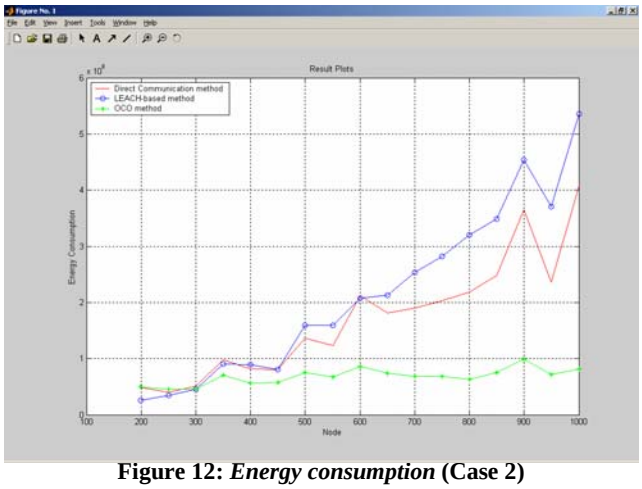


Figure 12: *Energy consumption* (Case 2)

The *accuracy* of OCO reaches 100% in almost all cases, as shown in Figure 13. LEACH's accuracy reaches 100% in cases when the number of nodes is greater than 750.

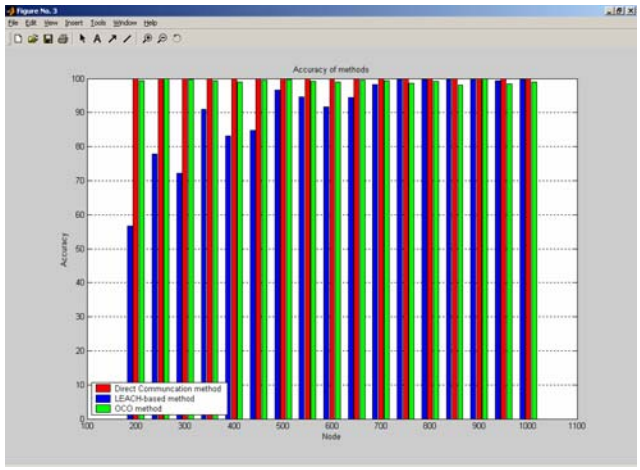


Figure 13: *Accuracy* (Case 2)

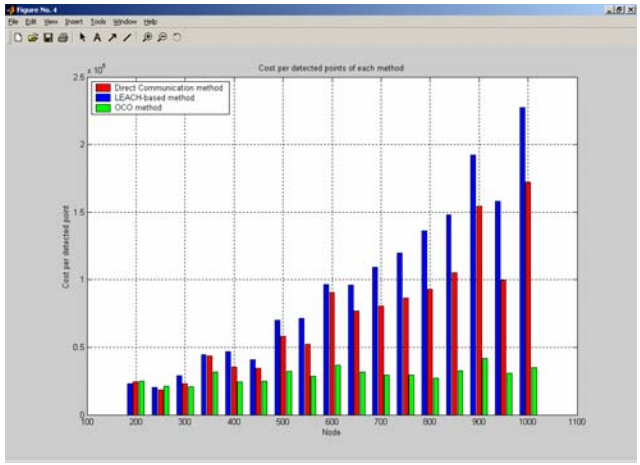


Figure 14: *Cost per detected point* (Case 2)

In Figure 14, OCO is shown to have spent the least energy with respect to each of the detected points, in almost all cases. In particular, at 800, OCO's cost is the lowest while ensuring almost 100% *accuracy* (Figure 13).

Similar to case 1, the results with respect to *time before the first dead node* (Figure 15) are somewhat 'fluctuating'. OCO still exhibits better longevity than LEACH across all cases.

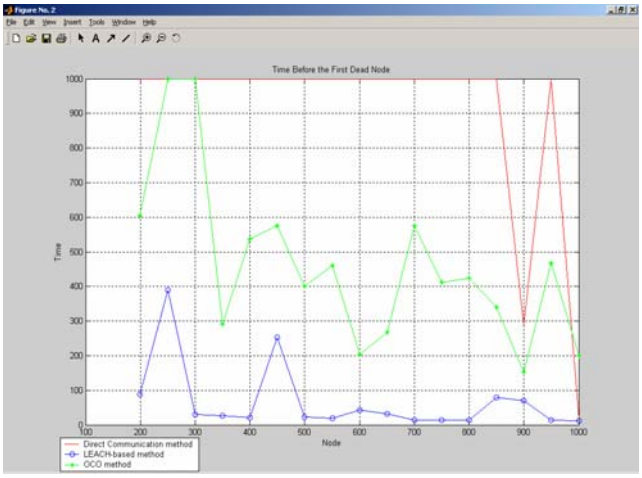


Figure 15: *Time before the first dead node* (Case 2)

5. SUMMARY AND FUTURE WORK

Based on our simulation study, OCO appears to consume less energy than the others, while maintaining maximum accuracy. One potential extension of our work is to evaluate the methods under different scenarios, for example with multiple intruding objects. In addition, we plan to add security features into OCO, since the sensor network usually operates in hostile environments.

6. ACKNOWLEDGMENTS

The authors are partially supported by the Univ. of Houston – Clear Lake (FRSF #859), the Institute for Space Systems Operations (ISSO), and the National Science Foundation (DUE 0311592).

7. REFERENCES

[1] Guo, Weihua, Zhaoyu Liu, and Guangbin Wu (2003). "An Energy-Balanced Transmission Scheme for Sensor Networks". *Dept. of Software and Information Systems - Univ. of North Carolina at Charlotte*. Retrieved 9/8/2005 at http://www.cens.ucla.edu/sensys03/proceedings/p300-guo.pdf.

[2] Heinzelman, Wendi R., Anantha Chandrakasan, and Hari Balakrishnan (2000). "Energy-Efficient Communication Protocol for Wireless Microsensor Networks". *Proc. of the Hawaii International Conference on System Sciences*, Maui, Hawaii. Retrieved 9/8/05 at http://academic.csuohio.edu/yuc/mobile03/0403-heinzelman.pdf

[3] Mallanda, C., A. Suri, V. Kunchakarra, S.S. Iyengar, R. Kannan, and A. Durresi (2005). "Simulating Wireless Sensor Networks with OMNeT++". *Dept. of Computer Science, Louisiana State Univ.* Retrieved 9/8/2005 at http://bit.csc.lsu.edu/sensor_web/final_papers/SensorSimulator-IEEE-Computers.pdf

[4] Pattem, Sundeep, Sameera Poduri, and Bhaskar Krishnamachari (2003). "Energy-Quality Tradeoffs for Target Tracking in Wireless Sensor Networks". *Dep. of Electrical Engineering and Dep. of Computer Science, Univ. of Southern California.* Retrieved 7/10/05 at http://www-scf.usc.edu/~pattem/PattemKrishnamachari_Tracking.pdf