Event Detection and Localization using Sensor Networks

R. Basnet, S. Mukkamala
Institute for Complex Additive Systems Analysis
Computational Analysis and Network Enterprise Solutions
New Mexico Tech, Socorro, NM, 87801

Abstract - *Although sensor network research was initially driven by military applications such as enemy tracking and surveillance in battle field, they have been deployed in several civilian and commercial applications such as habitat monitoring, environmental observations and forecasting, health monitoring etc. In this paper, we study and propose algorithms for detecting fire and localizing the same using sensor networks.*

Keywords: Sensor Networks, Event Detection, Event Localization

1 Introduction and Motivation

A pragmatic vision [1, 2] to deploy a large-scale, low power, inexpensive sensor network is possible thanks to recent advancement in Micro-Electro-Mechanical Systems (MEMS) [3]. In the past few years, a lot of efforts have been made to make this vision a reality [4]. Research prototype sensor nodes (UCB motes [5, 6], µAMPS [7], PC104 [8], etc.) are designed and manufactured, energy efficient MAC[9], topology control protocols [10, 11, 12] and routing schemes [13, 14, 15, 16, 17] are implemented and evaluated, various enabling technologies such as time synchronizations [18], localization and tracking [19] are being studied and invented.

Event detection may not be an interesting problem to solve using sensors. Event could be detected as simply as sensing some basic characteristics of the event. If the reading values of those characteristics cross certain thresholds, we could come to a rough conclusion that the event has occurred. Localizing the same event, however, is more challenging. If we could instrument the world with the sensors, localization would not be an issue. Though sensors are becoming more powerful and cheaper day-by-day as proclaimed by Moore's Law [20], it still is not feasible to place sensors everywhere. So, pin-pointing such an event, with a few numbers of sensors, has much more scopes in real-world applications such as in case of fire detection.

Appearing in *Proceedings of International Conference on Wireless Networks*, Las Vegas, Nevada, USA, 2009. Copyright 2009 by the author(s)/owner(s).

Modern smoke detectors are placed in households for the purpose of detecting fires [21, 22]. Most smoke detectors simply detect the presence of smokes and tend to generate too many false alarms while used for detecting fires [23]. Though most of the common smoke detectors can't recognize the traits of multiple fire hazards, there are smart fire alarms such as FireSmart [23] that is equipped with advanced software and neural network similar to human brain that can apply the data to distinguish fire from deceptive phenomena like vehicle exhaust, cooking fumes, humidity, cigarette smoke, dust, temperatures shifts, and radio interference from electronic devices. If we could detect and pin-point the origination of fire very accurately, we could, literally, save billions of dollars every year in loss and damages of properties due to the same. Once the fire's origination is localized, we could make the fire sprinkler system to focus on the area around that point to quickly and effectively extinguish the fire. The point of origination of fire could also be helpful in the aftermath for investigating on the cause of the fire.

The remainder of the paper is organized as follows. Section 2.1 outlines the leader selection methodology. Section 2.2 describes the algorithms and techniques for detecting fires. Section 2.3 outlines various algorithms and techniques we propose in localizing the fire. Section 3 discusses experiments and section 4 describes the results. Section 5 provides concluding remarks and future works.

2 Methodologies

2.1 Leader Selection

Among all the sensors placed in a given area, the leader node will play a key role in deciding fire and calculating its location or point of origination and triggering the fire sprinkler system if there's one. Each sensor is capable of being the leader node. The leader is dynamically selected based on the sensor data values. Once the data value for any properties: temperature, smoke, or light, has crossed the preset threshold in the sensor, the node becomes more alert and compares its own data values with the data received from the neighboring sensors. The sensor that has the highest data values will be elected as the leader node.

For instance, let's say node **A** has the following instance of data readings in its data table. The units of fire properties are irrelevant for the experiment, so we've used a range of real values between 0 and 100 for simplicity.

Table 1: An instance of data table in node A

Sensor ID	TimeStamp	Temp (100)	Smoke (45)	Light (100)
A	12/10/2005 10:14:42 AM	91	44	94
B	12/10/2005 10:14:49 AM	90	43	93
C	12/10/2005 10:14:47 AM	88	41	90
A	12/10/2005 10:14:53 AM	93.8	*46*	95.5

In Table 1, the number in the last three column headings shows the preset threshold value for that attribute. For example, smoke threshold is set to 45. The only data reading received from neighbor node B is: Temperature = 90, Smoke = 43, Light = 93. A has also received one data reading from neighbor C. Each row in the table is considered as one reading from that particular node. Two nodes are neighbors of each other if their wireless range are overlapping i.e., they can directly communicate with each other just in one hop. Since, A's current smoke value (46) has crossed the threshold value (45), it will be in alert mode and start checking and comparing its data with the data received from the neighbors. As we can easily see in fig. 1 that A has the highest current values for all the properties, A is the leader node. At this point, we know that the nodes B and C's data values have not crossed any threshold readings. So, in most cases, the leader node is self-elected. Then the elected leader node sends leader message with its current data values to the neighboring nodes. Most likely, the neighbors do not have their data values greater than the ones received from the leader node. However, the neighbor, for instance B, compares the received message with its data vales it received from its other neighbors say D and E. It is most likely that those neighbors will not have data values higher than what B has received from self elected node A. Thus, once the leader has been confirmed by all its neighbors, A will be the unanimously elected leader for that network. If all the nodes have similar data points (very unlikely if the sensors are placed very strategically), the node that initiated the leader message will assume the role of the leader. If all the nodes initiate the leader message at the same time (very unlikely again), we assume the situation is dire and the best thing to do would be to trigger the alarm and sprinkler system simultaneously.

2.2 Fire Detection

Detecting fire is trivial. For this purpose, I have used three fundamental characteristics of fire- heat, smoke, and light. When there is a fire, needless to say, there is some increase in temperature of the surrounding, there is some smoke in the area, and there is also some extra light. Motes have sensors that are capable of detecting these phenomena. The sensors will detect if there is any change (most preferably the increase) in temperature, smoke, and light of the area where the sensor is placed. If any change is detected in these characteristics beyond the *tct,* (user specified tolerance value); the change is noted, date-time-stamped and stored in local data table. The newly detected values are sent to the neighbors for localization which will be explained in next section. The readings that are sent to the neighboring sensors also have a unique identifier, SensorID to identify the sender of the data. In this fashion, the sensors keep recording the changes and pass back and forth the sensor-data among the neighbors.

Once the thresholds for temperature, smoke, and light are crossed, the node comes to a rough conclusion that the fire has occurred. To confirm, the node checks its current reading with the ones received from the neighboring sensors. If the neighboring sensors also have recorded the similar changes in the environment, the sensor unanimously decides that the fire has occurred in the area. If the node, that has detected the fire, doesn't have any record of the neighboring sensors' data, then it simply assumes that the neighbors have not yet sensed any changes in the environment. The node that detects and confirms the fire is most likely to be the leader as discussed in section *2.1*.

2.3 Fire Localization

Once the leader node detects and confirms the fire, then it becomes actively involved in calculating the location of the origination of the fire. I have used the following designs and assumptions to localize fire:

1. Geographical locations of a node and its neighbors are fixed and known.
2. A node will have its own data reading plus the readings it receives from its neighbors recorded in data table. Each data reading is analogous to a record in SQL database table. Table 1 shows an instance of a data table from a node. Each row in the table is one data reading recorded by a node with that Sensor ID.
3. If the change in at least one of the properties is noticed beyond *tct*, then the reading/data-row will consists of the current data values for all the properties.
4. The sensor nodes closer to the fire will start sensing the changes in environment such as temperature, smoke level, and light intensity earlier than the nodes farther away.

5. The fire will grow bigger with the time and it will grow pretty evenly with the growth in its radius.
6. At any instant of time, the node closer to the fire will have more count of readings than the nodes farther away. For the actual calculation of the location of the origination of fire.

We used different algorithms to calculate the location of fire. They are:

A. Single-Point,

B. Mid-Point,

C. Centroid, and

D. Count-based Refined Algorithm (CobRA)

A. Single-Point:
Single-Point algorithm says that the fire is closest to the leader node. So, basically the fire location is same as the location of the node. The algorithm will not work accurately if the sensors are widely distributed.

B. Mid-Point:
Mid-Point algorithm uses the mid-point of the leader and its neighbor. It has some problems. Since the leader node could have more than one neighbor, there would be four possible location of the fire. The nodes are placed in rectangular grid, so each node could have at most 4 neighbors. Only one mid-point will be closest to the fire location.

C. Centroid:
Centroid algorithm uses the leader's location and the location of two of its neighbors. It simply runs into problem when the leader node has more than two neighbors. There could be four different location of the fire. We need to come up with some approach to choose two neighbors and ignore the rest to get the better result with this algorithm.

D. Count-based Refined Algorithm (CobRA):
The term *count*, in this context, is the count of readings with selection based on the sensor ID. The SQL analogy of count would be: *COUNT* * FROM dataTable WHERE SensorID = neighborID.

CobRA has two steps. First, it uses the *count* to choose the nodes that will actively participate in calculating the location of fire. This is possible because each node has its own data readings plus the readings it received from the neighbors. Second, it uses any one of the three algorithms mentioned above based on the number of chosen neighbors. By using the *count*, the leader node will be able to select or ignore its neighbors. The neighbors with the higher *count* will be involved and the neighbors without any readings, 0 *count*, will simply be ignored. CobRA thus minimizes the area of interest and helps the leader node to choose one of the 3 algorithms for the best result.

Here are some of the case scenarios that explain what algorithm to use in what circumstances:
Case I: If the *count* of a neighbor is 0, ignore it. If the *count* of all the neighbors is zero, use Single-Point algorithm. This case assumes that the event has occurred very close to the leader node and its neighbors have not sensed any changes in their readings yet.

Case II: If the leader node has more than 2 neighbors, consider only 2 that have the higher *count.*

Case III: If two neighbors have the same *count*, use Mid-Point between these and again Mid-Point of that mid-point and the leader's point.

Case IV: If the two neighbors have some *count*, use the Centroid of three points.

Mid-Point algorithm can further be refined based on the difference in *count* of the two nodes. The calculated location of the event will be further moved proportionately closer to the node that has the higher *count*.

3 Experiments and Results

3.1 Simulation Environment

To study the effects of the proposed algorithms, we used iSIM, iPAQ Simulator, developed by University of Oregon [24]. The iSIM simulates a floor of a building with several rooms, corridors, doors, furniture, etc. It basically allows us to place person, fire, access points, robots, etc on any part of the floor. We added sensor nodes such as fire sensor, fire sprinkler system, motion detector, etc. to the simulator. The simulated 2-D environment consisted of a fairly big size rectangular room with strategically placed 4 fire sensors; one on each corner of the room. Fire sprinkler system, controlled by fire sensors, was placed in the center of the room to extinguish the fire. The following figure 1 may give some visual aspect of the simulated room.

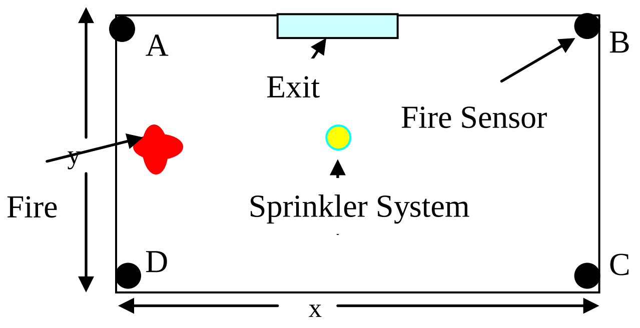

Figure 1: A Smart Room

The sensor nodes in the room communicate with each other through server/client sockets. All the fire sensors have exactly the same characteristics. The sensors have capability to sense the characteristics of fire up to about half he breadth of the room. Sensors on the diagonal to each other were not the neighbors of each other, for instance.

3.2 Results

We ran the simulation for about 6 times placing the fire on each side of the room and close to the center hoping to cover most part of the room and to make the event as a random phenomenon. We then compared the accuracy of the results given by each algorithm. The average percentage error is the average error in X and Y co-ordinates. The error in X co-ordinate is calculated first and then the error in Y co-ordinate. The average of these errors is then taken into account for the comparison purpose. In the legend of the charts, Mid-Point I is the mid-point between the leader and the first neighbor, Mid-Point II is the mid-point between the leader and the second neighbor, and Mid-Point III is the mid-point between the two neighbors of the leader. The Single-Point, Centroid, and CobRA algorithms are as explained before.

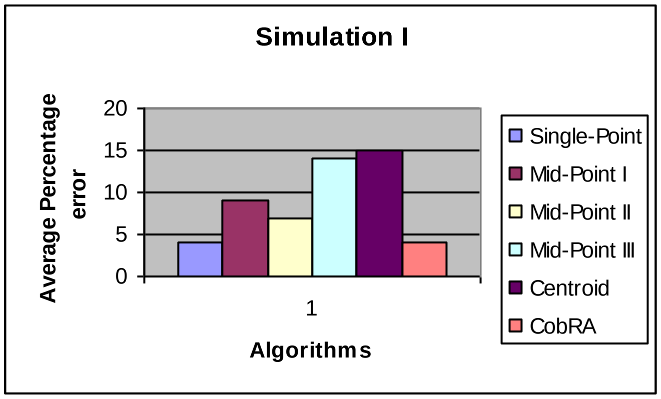

Figure 2: Experiment I

In Experiment I (Figure 2), fire was placed closed to the node A. CobRA selected the Single-Point algorithm.

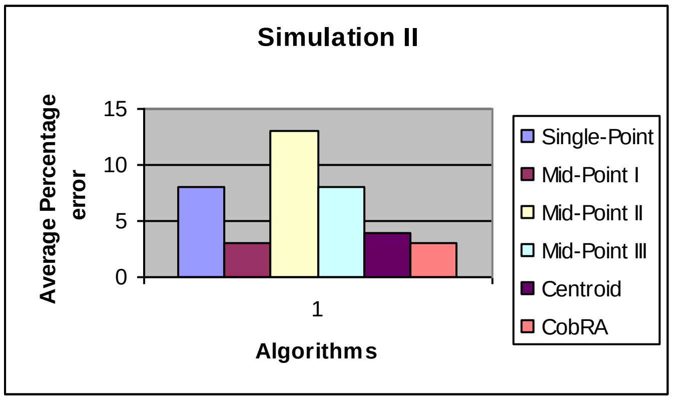

Figure 3: Experiment II

In Experiment II, fire was placed somewhere in between node A and node D, CobRA selected Mid-Point algorithm.

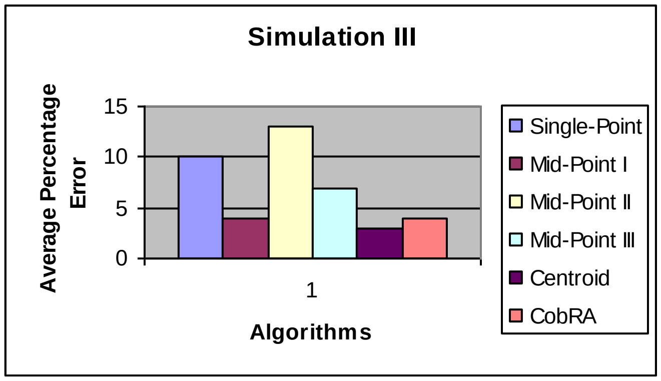

Figure 4: Experiment III

In Experiment III (Figure 4), the fire was placed little above the center for the room. The CobRA selected the Mid-Point algorithm between node A and D.

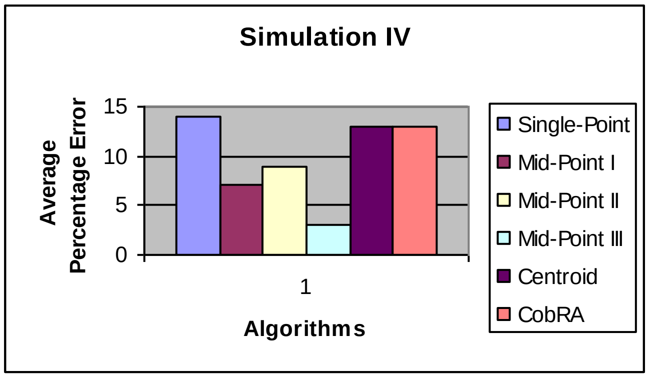

Figure 5: Experiment IV

In Experiment IV (Figure 5), the fire was placed close to the center of the room. The CobRA selected Centroid algorithm. All four nodes had some *count* values when the leader node A calculated the location of the event. The leader A used its neighbors B and D to calculate the location of the fire using Centroid algorithm.

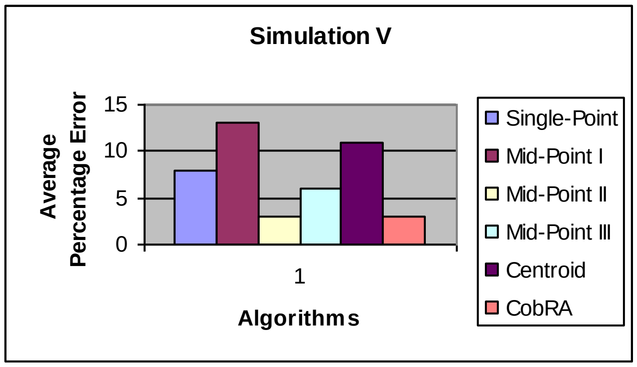

Figure 6: Experiment V

In Experiment V, the fire was placed some where closer to C and D. The CobRA selected Mid-Point between C and D. In Experiment VI, the fire was placed very close to in-between node C and D. The CobRA selected Mid-Point

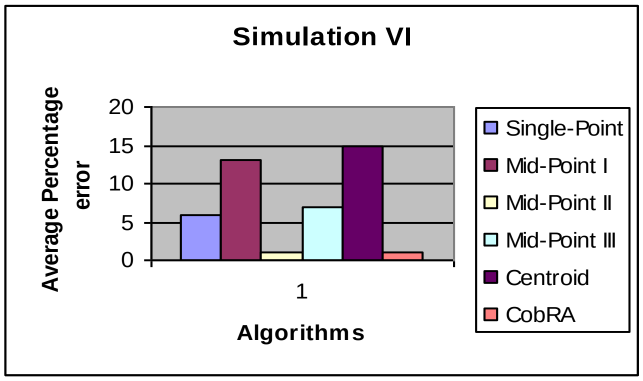

Figure 7: Experiment VI

algorithm. The best algorithm has about 1% error because the fire was almost exactly in-between node C and node D.

4 Conclusion and Future Work

Out of four algorithms proposed, CobRA seemed to work the best in most cases. CobRA did not give the better result only when the fire originated somewhere in the center of the room. Single-Point, Mid-Point, and Centroid algorithms do not seem to work well by themselves. Moreover, they pose ambiguity in choosing the neighbors to calculate the location of the event. CobRA seemed to work well in selecting the neighbors and then selecting one of the first three algorithms to localize the event. The experiment was done on a 2D simulation, which obviously raises some questions on how the proposed research could be applied to 3D world. While we think that it would be simply a matter of adding another dimension, height to the mid-point co-ordinate calculations, we leave the actual experiments and results for future work.

5 References

[1] D. Estrin. Embedded networked sensing research: Emerging system challenges. In NSF Workshop on Distributed Communications and Signal Processing. Northwestern University, December 2002.

[2] D. Estri. R. Govindan, J. Heidemann, and S. Kumar. Next century challenges: Scalable coordination in sensor networks. In Proc. ACM/IEEE MobiCom, pages 263-270, 1999.

[3] What is MEMS Technology? http://www.memsnet.org/mems/what-is.html. Accessed on February 9, 2009.

[4] Ning Xu. A Survey of Sensor Network Applications. University of Southern California.

[5] J. Hill and D. Culler. A wireless embedded sensor architecture for system-level optimization. Technical report, Computer Science Department, University of California at Berkeley, 2002.

[6] J. Hill, R. Szewczyk, A. Woo, S. Hollar, and D.C.K. Pister. System architecture directions for networked sensors. In Proceedings of ACM SIGMOD, San Diego, CA, June 2000.

[7] µAMPS Home. http://www-mtl.mit.edu/researchgroups/icsystems/uamps/. Accessed on February 9, 2009.

[8] Getting Started with PC-104 Testbed. http://www.isi.edu/scadds/pc104testbed/guideline.html Accessed on February 09, 2009.

[9] J. Heidemann, W. Ye and D. Estrin. An energy-efficient mac protocol for wireless sensor networks. In Proceedings of the 21st International Annual Joint Conference of the IEEE Computer and Communications Societies (InFOCOM 2002), New York, NY, June 2002.

[10] J. Heidemann, Y. Xu and D. Estrin. Geography-informed energy conservation for ad hoc routing. In Proceedings of the Seventh Annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom 2001), Rome, Italy, July 2001.

[11] Y. Xu et al. Topology control protocols to conserve energy in wireless ad hoc networks. Technical Report 5, University of California, Los Angeles, Center for Embedded Networked Computing, January 2003.

[12] H. Balakrishnan et al. SPAN: An energy efficient coordination algorithm for topology maintenance for in ad hoc wireless networks. In Proceedings of the Seventh Annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom 2001), July 2001.

[13] C. Intangonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In International Conference on Mobile Computing and Networking (MOBICOM), August 2000.

[14] D. Braginsky and D. Estrin. Rumor routing algorithm for sensor networks. In WSNA, September 2002.

[15] B. Karp and H. T. Kung. Gpsr: greedy perimeter stateless routing for wireless networks. In MOBICOM, 2000.

[16] R. Govindan et al. The sensor network as a database. Number TR02-02-771, September 2002.

[17] R. Govindan et al. Data-centric storage in sensornets. In WSNA, 2002.

[18] J. Elson, L. Girod, and D. Estrin. Fine-grained network time synchronization using reference broadcasts, 2002.

[19] H. Wang et al. A wireless time-synchronized cots sensor platform part ii – applications to beam forming. In Proceedings of IEEE CAS Workshop on Wireless Communications and Networking, Pasadena, CA, 2002.

[20] G. E. Moore. Cramming more components onto integrated circuits. Electronics, Volume 38, Number 8, April 19, 1965.

[21] Fire in the Home. http://www.usfa.dhs.gov/downloads/pyfff/inhome.html. Accessed on February 9, 2009.

[22] USFA Smoke Alarms. http://www.usfa.dhs.gov/citizens/all_citizens/home_fire_prev/alarms/. Accessed on February 10, 2009.

[23] FireSmart Fire Detectors. http://www.faradayfirealarms.com/pdf/FireSmartBroch.pdf. Accessed on Feb 11, 1009.

[24] iSIM: iPAQ Simulation Environment. http://www.cs.uoregon.edu/research/wearables/iSIM/