Tracking Moving Targets in a Smart Sensor Network

Rahul Gupta
Department of Electrical & Computer Engineering
and Computer Science
University of Cincinnati
Cincinnati, OH 45221-0030
and
Samir R. Das
Computer Science Department
SUNY at Stony Brook
Stony Brook, NY 11794-4400

Contact Email: samir@cs.sunysb.edu

Abstract

Networks of small, densely distributed wireless sensor nodes are capable of solving a variety of collaborative problems such as monitoring and surveillance. We develop a simple algorithm that detects and tracks a moving target, and alerts sensor nodes along the projected path of the target. The algorithm involves only simple computation and localizes communication only to the nodes in the vicinity of the target and its projected course. The algorithm is evaluated on a small-scale testbed of Berkeley motes using a light source as the moving target. The performance results are presented emphasizing the accuracy of the technique, along with a discussion about our experience in using such a platform for target tracking experiments.

1 Introduction

Rapid advances in miniaturization in computing and sensor technologies and advent of low-power short-range radios recently have given rise to strong interest in smart sensor networks [8, 7]. The idea is to be bring together sensor nodes with on-board processing capability and radio interface into a large network to enable them to process higher level sensing tasks in a co-operative fashion. Several new design themes have emerged for such networks. For example, the network must be fully self-configuring and highly fault-tolerant as the sensors may be deployed in an "ad hoc" fashion. The network must minimize battery power usage; this enables untethered and unattended

operations for an extended time. A corollary of the latter property is that the system must leverage data processing and decision making ability inside the network as much as possible, instead of shipping the data to a central controller to make decisions. This is because with current day technology, the power budget for communication is many times more than that for computation.

An emerging application area for smart sensor networks is intelligent surveillance or monitoring. Sensors are distributed, likely randomly, in a geographic area to be monitored. The goal is to track and predict the movement of an appropriate target and alert the sensors which are close to the predicted path of the target. The target can be a moving vehicle, for example, or can be a phenomenon such as an approaching fire. It is assumed that each individual sensor node is equipped with appropriate sensory device(s) to be able to detect the target as well as to estimate its distance based on the sensed data. The sensors that are triggered by the target collaborate to localize the target in the physical space to predict its course [13]. Then the sensor nodes that lie close to the predicted course of the target are alerted. This alert is meant to serve as a trigger for these nodes to activate additional on-board sensors. For example, these additional sensors may be of a different modality (e.g., alerts coming from heat sensors activating vibration sensors) that are ordinarily turned off or not sampled to conserve power. The alert can also serve as a trigger to actuate certain on-board devices, depending on the capability of the nodes and the application.

The goal of this paper is to develop techniques for the above moving target tracking problem and report our experience in testing them in a live low-cost sensor network testbed using Berkeley motes [11]. Variations of these motes have been used in several experimental testbeds recently. See, for example, [6, 14, 9, 16]. We will demonstrate the feasibility of our approach. We will also discuss performance results, and several practical problems a designer/implementor must be aware of.

The paper is organized as follows. In the next section we describe the hardware and software architecture of the sensor nodes used. The tracking algorithm is described in Section 3, and experimental evaluation is reported in Section 4. In Section 5, we discuss the problems we faced in our experiments and some ways to alleviate them. We conclude in Section 6.

2 Sensor Network Testbed

Our testbed comprises of 17 Berkeley motes [11] based on the MICA platform and manufactured by Crossbow technology [2]. For sake of completeness, we briefly review the hardware and software architecture of these motes.

2.1 Hardware Overview

In order to create a low-power system, MICA platform is based on a single central microcontroller that performs all sensing, communication and computation tasks. MICA motes use ATmega 128L processor by Atmel [1], that is driven by a 8 MHz external crystal and has a 8-bit data bus. It

has 128 KB of program memory and 4 KB of data SRAM. There is a 10-bit internal ADC that is directly connected to the sensors. Hence a 10 bit reading is obtained from the sensors in the range 0-1023. This will later be used to measure sensor signal strength.

The RF module consists of an RF Monolithics 916.50 MHz transceiver (TR1000) [3]. It can be externally controlled, through a potentiometer, to have a communication radius ranging from a few inches to tens of yards, and can operate at data rates up to 115 kbps. A key characteristic of the radio is that it only consumes 12 mA while transmitting and 5 mA while receiving. An antenna for the radio has been integrated into the surface of the printed circuit board and connecting an external antenna is optional. The system is designed to operate off a pair of AA batteries.

The medium access protocol used is a variant of the Carrier Sense Multiple Access (CSMA) protocol [17]. There is a random delay before the transmission of every packet. If the channel is busy, the node backs-off for a random amount of time. During backoff, the radio is powered off to save energy and no communication is possible during this period. This MAC protocol does not have maximum number of backoffs, and keeps trying until a clear channel is found.

The I/O subsystem consists of a 51-pin expansion connector. It helps in interfacing the microcontroller with the sensing board. This connector is also used to program the microcontroller by connecting it to the programming board, which in turn is connected to the parallel port of the PC. The connector also provides a serial port interface to communicate with the MICA mote and hence can be used to transfer data to PC and debugging. Three LEDs present on the board act as actuators that can be controlled through software and act as a user interface.

2.2 Software Architecture

The key requirements of an operating system that run on resource-constrained sensor nodes are (i) small memory footprint, and (ii) effective management of hardware in terms of power consumption and processing time.

MICA motes run on an event-based operating system, called *TinyOS* [11]. TinyOS fits in 178 bytes of memory. It manipulates the hardware directly and there is no kernel layer. To avoid the overhead associated with context switch and process management, only one process can be active in the system at one time. There is no dynamic memory allocation and the memory is allocated at compile time. The TinyOS code and applications are compiled together and run in a single linear address space. This reduces the memory management overhead. TinyOS is divided into a collection of software *components*. The complete system software consists of a scheduler and an interconnection of these components. An application may consist of a number of layers of these componenets stacked on top of each other. There are three types of components:

- Hardware Abstraction Components: These components directly map to a physical device (such as LEDs, UART, ADC) and are used to manipulate them.
- Synthetic Hardware Components: These components simulate the behavior of hardware, in

case the hardware is absent in the mote.

- High Level Components: These components perform various data manipulations and transformations, such as a routing algorithm or any other application.

TinyOS provides two levels of scheduling - *Events* and *Tasks*. This helps to do all the processing in real-time. Events are synchronous in that they are serviced to completion as soon as they occur. This includes interrupts from the hardware. Events cannot be preempted. Tasks are asynchronous and involve time-consuming computations. Tasks are managed by a scheduler that schedules these tasks when no event is to be processed. Events can preempt a task.

For communication, a fixed size of 38-byte packets are used in TinyOS, with 30 bytes of payload. Message header includes a 16-bit CRC error checking. A mote is uniquely identified by a 16-bit ID that is assigned while uploading the application code to the mote. Each message contains the destination mote ID. Two special IDs - 0xffff and 0x7e - are reserved for broadcast and UART, respectively. A message destined for UART is sent to the serial port.

A message-based model called Active Messages [15] is used by MICA motes to communicate with each other. Each message contains the name of a user-level handler to be invoked on the target mote, and a data payload that is passed as arguments. If no message handler is intended for the message, the message is discarded without further processing.

3 Tracking Moving Targets

We assume that the sensor nodes are scattered randomly in a geographical region. Each node is aware of its location. Location information can be gathered using an on-board GPS receiver. Absolute location information is, however, not needed. It is sufficient for the nodes to know their location with respect to a common reference point. Many localizing techniques can be used with varying degree of hardware complexity and accuracy. See, for example, [16, 5]. The sensor nodes are stationary in our model; this makes the localization problem somewhat simpler. Since the work presented here is not dependent on any particular localization method used, we do not emphasize any particular technique. In the experiments reported, we have directly encoded the location information into the sensor nodes to eliminate the possibility of any localization error.

The sensors must be capable of estimating the distance of the target to be tracked from the sensor readings. It is assumed that the sensor has already learned the sensor reading to distance mapping. We conducted a separate set of experiments to determine this mapping and encoded the mapping directly as a table in the application component.

Tracking a target involves three distinct steps:

1. Detecting the presence of the target.

2. Determining the direction of motion of the target.

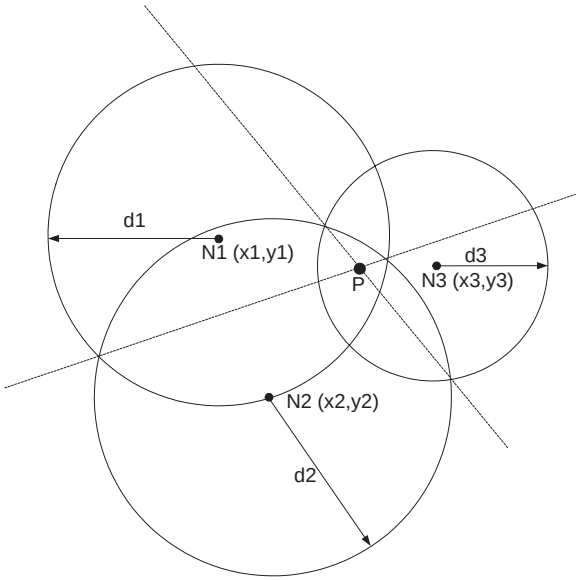

Figure 1: Three distance estimates d_1, d_2 and d_3 at nodes N_1, N_2 and N_3, respectively, with known location coordinates result in three circles. Two straight lines are drawn through the points of intersection of two pairs of such circles. The target is localized at the intersection (P) of these two straight lines.

3. Alerting appropriate nodes in the network.

These steps are discussed in detail in the following subsections.

3.1 Detection

Each node periodically (every 1 sec in our experiments) polls its sensor module to detect the presence of any target to be tracked. Sensor reading above a particular threshold indicates the presence of a target in the vicinity. As soon as this threshold is crossed, a *TargetDetected* message is broadcast by the node. Each *TargetDetected* message contains the location of the originating node and its distance from the target, as determined from the sensor reading. When this message is received by a neighboring node, it stores the coordinates of the originator and the target's distance from the originator in a table. Table entries expire after a timeout (4 sec in our experiments) unless refreshed.

3.2 Tracking

The next step is estimating the location of the target. A minimum of three nodes sensing the target are needed to apply the commonly used triangulation method [12]. See Figure 1 for an explanation of how we used it. When a node that has already detected the target hears two additional *Target-Detected* messages from two different neighbors, it computes a location estimate via triangulation. Note that any node that hears three *TargetDetected* messages from three different neighbors can estimate the location of the target. However, we limit this computation only to the nodes that

themselves have detected the target, and hears from two other neighbors that also detected the target. This limits the estimation to be done only in nodes within a close vicinity of the target, thus localizing the computation.

In order to estimate the trajectory of the target, its location must be estimated at a minimum of two instants of time. A straight line through these two points defines the trajectory in the direction of the latest location estimate. We found that with only two estimates, the impact of any estimation error was significant. Three or more estimates, however, worked significantly better. In the experimental results that follow we used three estimates with linear regression to compute a best-fit straight line. This line defines the estimated trajectory of the target. Note that more estimates, along with a higher-order curve fitting, will improve accuracy further. More estimates, however, will require a larger network to experiment with. For better accuracy, location estimates are used for trajectory estimation only when they are separated by at least a minimum distance (3 inches in our experiments).

3.3 Alerting

After estimating the trajectory, the network must alert nodes that lie near the trajectory (specifically, within a perpendicular distance d from it, d being 5 inches in our experiments) by sending them a *Warning* message so that they are aware of the approaching target and can take appropriate actions. Any node that is able to estimate the trajectory by using three location estimates broadcasts a *Warning* message. The message contains the location of the sender and parameters describing the equation of the straight line trajectory. Any node receiving the *Warning* message rebroadcasts it, if it is located within a distance d from the trajectory.

Care must be taken to prevent propagation of this warning message in the direction opposite to the direction of motion of the target. This is done via some simple geometric considerations. The node receiving a *Warning* message computes, through itself, a line perpendicular to the trajectory. The line divides the geographic area into two regions R_1 and R_2, R_2 being towards the direction of motion. A node forwards the *Warning* message if (i) it lies within a distance d from the trajectory, and (ii) the *Warning* message is received from a node in region R_1. See Figure 2. This ensures that only the nodes within d distance from trajectory and towards the direction of motion forward the warning. This localizes message propagation only in the relevant part of the network.

Note that the above technique assumes that the network has enough density such that the subset of the sensor network nodes, that lie in the region where warning message must be propagated, must form a connected graph among themselves. This condition is needed as the warning message propagation is suppressed outside this region. Without this assumption, simply larger regions need to be flooded with warning messages.

Note that multiple nodes may originate *Warning* messages for the same detected target. This is because any node detecting an target independently attempts to carry out location and trajectory estimations. To conserve bandwidth and power, we stipulate that a node refrain from forwarding

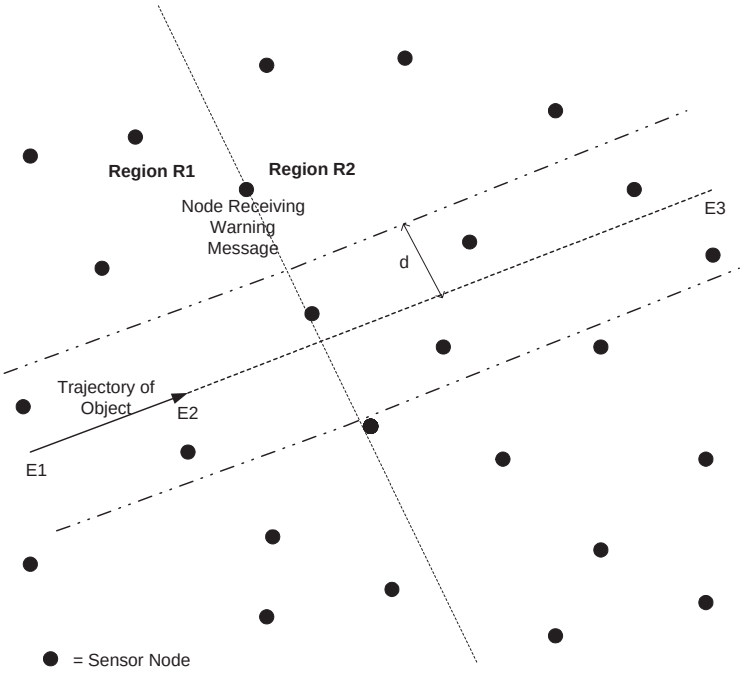

Figure 2: Sensor network with a moving target. $E_1 - E_2 - E_3$ defines the trajectory of the moving target.

a *Warning* message for some time (20 sec in our experiments) after it has forwarded one. This also implicitly assumes the presence of only one source in the network at any time. Detecting the presence of multiple sources and tracking them on an individual basis will require sophisticated sensing and signal processing algorithms [13] that is beyond the scope of our current work.

4 Experimental Evaluation

In our experiments the moving target is a light source (bulb of a flashlight, taken out of the casing to minimize shadows and operated using four AA batteries). The experimental platform is a 60 inch $\times$ 60 inch square area with 16 motes placed at random locations. Another mote is used as a probe to capture all packets transmitted in the network for debugging and tracing functions. The probe does not participate in the algorithm. Recall that due to the absence of any localization system, the locations are encoded in the motes before the start of the experiment.

4.1 Characteristics of the Photo Sensor

We first performed a set of experiments to determine the relationship of the sensor reading with the distance of light source. This relationship would later be used to estimate the target distance from sensor readings. We faced several complications here. First, different sensors generated different

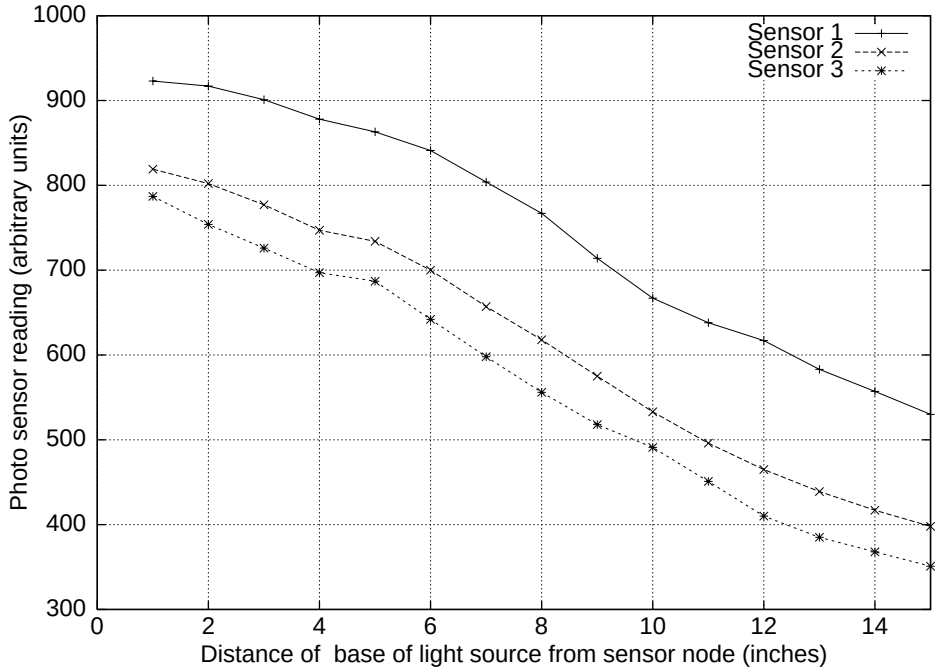

Figure 3: Relationship of the "average" sensor readings with the distance of the light source. Three different sensors are shown to point out the difference in sensor characteristics.

readings for the same distance of the light source. The readings was variant enough that we felt some calibration would be necessary to reduce errors. While statistical methods using parameter estimation techniques such as reported recently in [16] could be used, we chose to determine the exact sensor reading versus distance relation for *each* individual sensor. This was feasible as we were dealing with a small number of sensors. Second, we found that light falling on the photo-sensor at an angle made the reading sensitive to the direction of the light source relative to the sensor. To reduce this sensitivity, we experimented with the light source at an elevated plane (at a height of 9 inches). This also resolved shadowing problems at small heights caused by the hardware components on the mote. The light source was always kept at the same height as we are interested in solving the tracking problem in two dimensions only.

Figure 3 shows the average sensor reading vs. distance plots for three different sensors. Notice the differences in sensor characteristics. We tested all individual sensors and encoded the sensor reading versus distance function in their application components.

4.2 Target Tracking

Experiments are performed by randomly placing 16 motes in a 60 inch $\times$ 60 inch area in a dark room. A small area is chosen intentionally so that the experiments can be performed on a table top; this keeps the experiments manageable. A threshold of 15 inches is used for the distance of the light source; beyond this distance the sensor reading is assumed too low to be reliable. The target

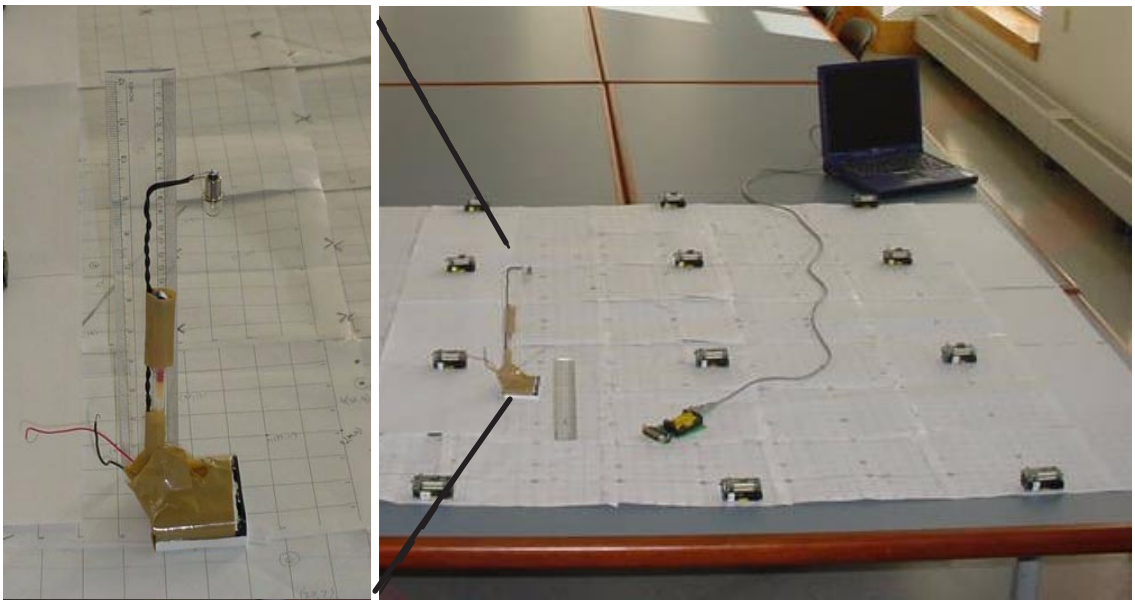

Figure 4: The sensor network setup, showing the probe node connected to the laptop and the elevated light source.

must be closer than this distance for a sensor to be able to detect it.

A probe mote is kept inside the experimental region such that it can hear packets transmitted by *all* motes in the network. This probe is used to monitor all network traffic, and used for debugging and tracing activities. The probe passively listens to all transmissions and does not transmit anything. The packets gathered by the probe are transmitted via a serial interface to a laptop computer. See the picture of our experimental setup in Figure 4. Many random placements of sensors are used as test cases; however, because of space limitations results from only a few placements are reported here.

In the first set of experiments, we evaluate the location estimation error. To evaluate the error, the light source is placed at approximately 5 inch intervals spanning the whole region in a grid-like fashion, and its location is estimated by three sensors exactly as described previously. The error (i.e., distance) between the actual and estimated location is computed. The average and standard deviation of the error is presented in Table 1. Note that the average error is small (less than 2.5 inches), while the standard deviation of the error is relatively high. This is due to the fact that the error due to orientation of the light source relative to the photo sensor is not completely eliminated even with the elevated light source. A secondary reason is the inherent variability of the sensor data.

The next set of experiments evaluates the performance of the tracking algorithm itself. Here, for each random placement of the motes, we perform a series of experiments where the light source is dragged slowly along a straight line path in the experimental region. The network responds by predicting the trajectory and lighting the LEDs in the motes which receive *Warning* messages. A

Placement	Avg. estimation error (inches)	Std. dev. of error (inches)
1	2.32	1.82
2	2.06	1.23
3	2.23	1.53

Table 1: Location estimation error statistics for three different random mote placements.

large number of experiments are performed; but only a few are reported here for brevity. The nature of these experiments is such that it is hard to present the results in a quantitative fashion for readers to get a fair idea of the performance. So we have chosen to present the results in a visual fashion in Figures 5 through 8. These figures are automatically generated by a script running on the trace of the packets gathered by the probe node and transmitted to the laptop. They are explained below.

Small rectangles indicate the position of sensor nodes. The light source representing the moving target is actually moved from point P_1 to point P_2 in the experiments, indicated by a line. Circles around some nodes indicate the distance at which the light source is being sensed by the corresponding nodes. Circles are drawn only for the last time any node "sees" the source and sends out *TargetDetected* message. The line through E_1 and E_2 (in the direction of E_2) denotes the estimated trajectory. Thus, the difference between P_1P_2 and E_1E_2 denotes the estimation error. The following notations are used to designate nodes participating in warning message propagation:

- *"WO"* denotes the node that originates the *Warning* message.
- *"WR"* denotes the node that receives a *Warning* message and lies within distance d of the estimated trajectory, but does not forward the message because it lies in the direction opposite to the direction of motion.
- *"WF"* denotes the node that receives a *Warning* message, lies within distance d of the estimated trajectory and also forwards the message. WF nodes are highlighted by drawing a rectangle around them.

As mentioned before, three different placements of motes are reported. For each of these placements, the light source is slowly moved from point P_1 to point P_2. While many experiments were performed with different placements of motes and different movement paths of the light source, for brevity only three sample results are shown in Figures 5 through 7. In Figure 5, note that even though several motes originate a warning message, they all compute the same path. In Figure 7, note that the predicted direction of motion is somewhat off from the actual direction. This is because of a large error in one or more of the three location samples that are used in estimating the trajectory. The accuracy of the prediction can be improved naturally by taking many more

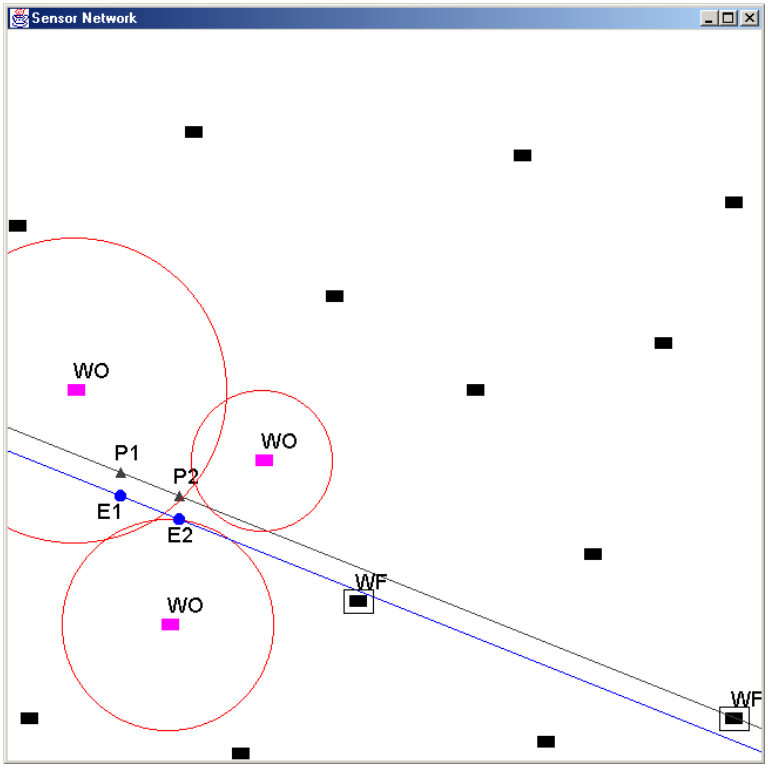

Figure 5: Tracking moving target: experimental scenario 1.

samples, which will, however, require "observing" the target (light source) for longer time. This will also require larger experimental area and a larger number of motes.

We have noted several interesting scenarios in the cases we studied. One example is presented in Figure 8 where different trajectories are estimated by two originating nodes, because they "heard" different sets of nodes that detected the target. The accuracies of the two estimates are very different. We looked carefully into the traces of this scenario, and found that the location estimation errors are not very high (maximum 3 inches); but biased errors for one estimation resulted in a very different trajectory. Once again, larger number of samples with a larger network should improve such situations significantly. The second problem in Figure 8 is that no warning message is ever propagated. The reason is simply that there are no sensors within distance d of either of the predicted trajectory. This situation will trivially improve by choosing a denser network to experiment with or choosing larger value of d.

5 Discussions

It is worthwhile to outline here briefly the problems we faced in our experimental work using low-cost sensor nodes, not capable of sophisticated signal processing.

The photo sensor used in MICA motes is sensitive to the angle at which light rays are incident on the sensor. This makes estimating distance from sensor readings hard, as the angle influences

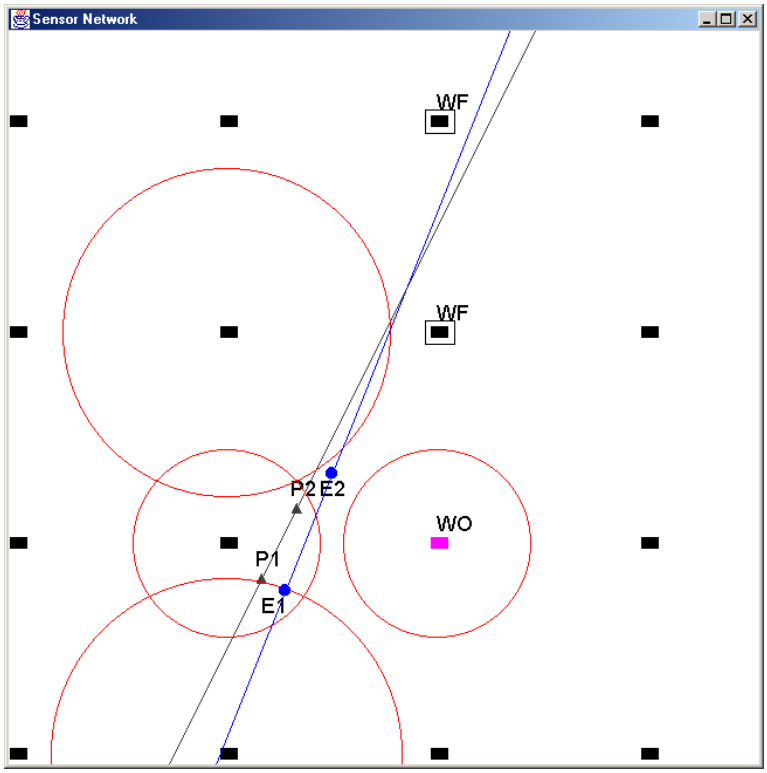

Figure 6: Tracking moving target: experimental scenario 2.

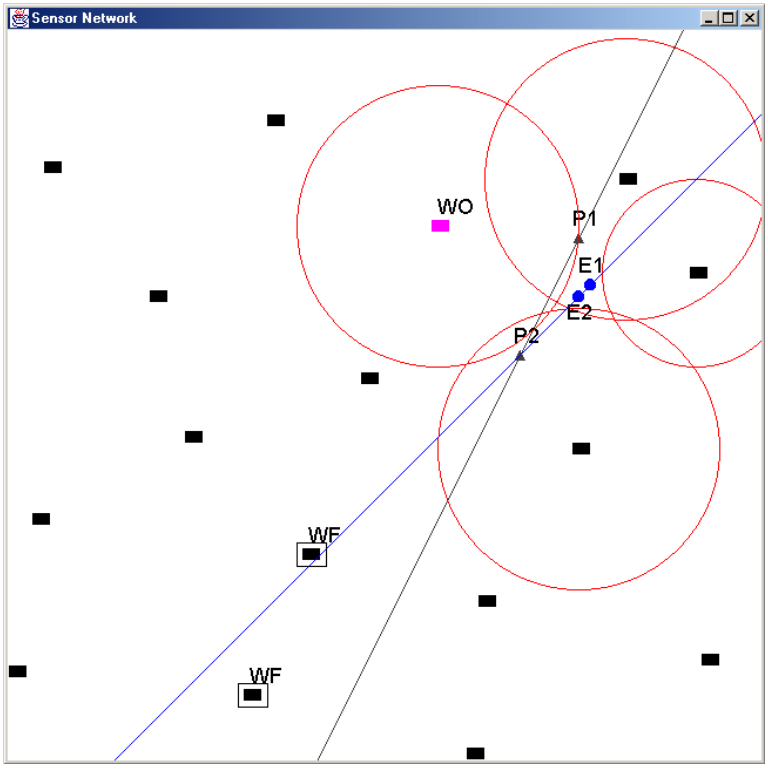

Figure 7: Tracking moving target: experimental scenario 3.

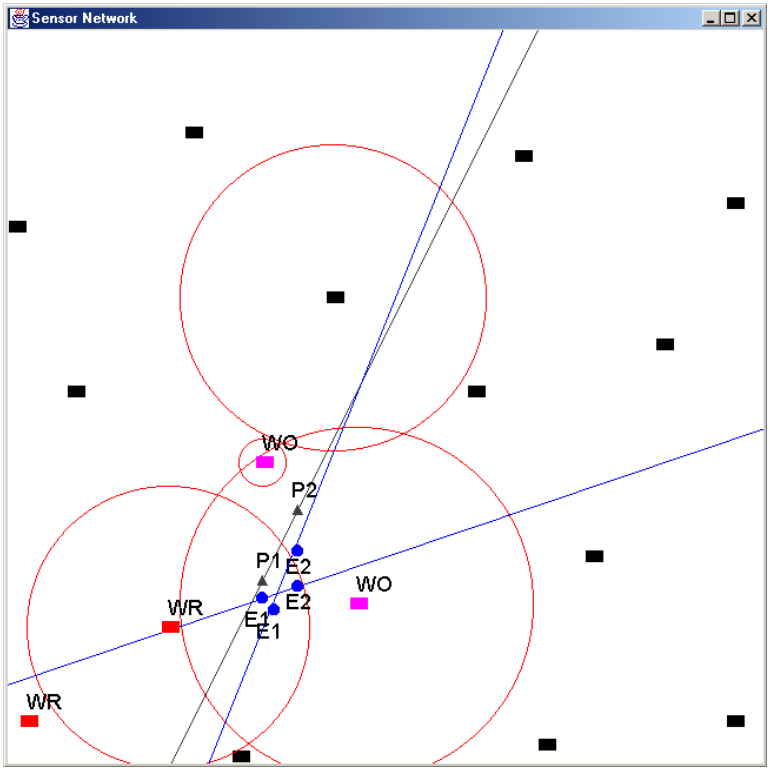

Figure 8: Tracking moving target: an interesting experimental scenario.

the reading. We also noticed that very small changes in the distance influenced sensor readings significantly when the light source is near the sensor. This is because relative changes in the angle is more significant at closer distance. An elevated source of light reduces this problem, but does not completely eliminate it – one reason for large variations in the errors in location estimates reported in Table 1. In addition, it was hard for us to procure a light source which is truly omnidirectional in nature. We observed variations in sensor readings in different directions even when elevation (and hence angle) and the distance of the light source were kept constant. We conjectured that either the light source or the sensor (or both) have some directional properties, which we ignored and relied on *average* properties to predict distance from sensor readings. While these observations are particular to light source and photo sensors on MICA motes, we believe some of these problems will also confront designers when other signals and sensors (e.g., acoustic or magnetic) are used.

To ensure that the light source always emits light with the same power, we used freshly recharged batteries for all experiments. While this is possible for controlled experiments as reported here, it will be impossible in real scenarios. So we conjecture that estimating distance of the target from a set of sensor readings will be difficult to impossible in general settings, as the source signal cannot be always expected to be of a consistent strength. Unless the sensors are very sophisticated, they will at best be programmed to simply detect presence or absence of a signal (and not to estimate any distance), and then collaborate with neighboring sensors to increase the confidence that an event or phenomenon has occurred in the vicinity. The strength of the signal can

be gathered indirectly by determining how many sensors can detect the signal and how spread out they are geographically. We feel that this would be a reasonable approach to pursue, which will not be dependent on omnidirectional signals of consistent strength. However, this simple technique, for all practical usage, will require a large number of densely disposed sensors. This will be a direction we plan to pursue in our future work.

We had to calibrate each sensor individually, so that the variability introduced by the manufacturing process that influences their sensitivity does not impact our experiments. This will be hard to do for a very large number of sensors. Thus, sophisticated statistical methods such as reported in [16] will need to be adopted.

In our experiments we have directly encoded the location of the sensor in its program. But in real applications, they need to be localized. While many methods have been reported in recent literature [5, 10, 16] – some of which applicable to MICA motes – they will all introduce their own sources of error. A statistical analysis of errors has been done in [16]. It will be interesting to analyze the combined effect of two types of errors – localization errors of the sensor themselves and errors in location estimates and trajectory computation of the moving target.

As discussed previously, a large number of location samples (we used only three in our experiments) has a strong potential to reduce errors in estimating the trajectory. However, this will require a larger testbed. Also, the times at which different sensor nodes are sampling the signal are not synchronized. Thus, errors could be introduced when signals sampled at different times are combined for locating the moving target, as the actual location of the target could change. This error can be minimized by sampling sensors at a much higher rate relative to the maximum speed of the target. Of course, time synchronization can also be introduced at the cost of higher design complexity or possible power usage.

6 Conclusions

We have developed a simple algorithm for tracking moving targets in a smart sensor network. The sensor nodes detect and track the moving target in a collaborative fashion and alert the nodes near the predicted path of the target. The algorithm localizes the communication in the vicinity of the location of the target and its estimated trajectory. This is critical as the sensor nodes usually run on a low power budget. The strength of our work is that we implemented and evaluated the algorithm in a real sensor network testbed using Berkeley motes. We used a moving light source as target. We described several factors that influence the accuracy of target tracking using low-cost sensor nodes such as the motes, and the potential problems a designer can face. The accuracy in our experiments has been fairly good, but not excellent. The accuracy is greatly influenced by the number of location estimation samples the designer can work with. With a small number of motes and a small experimental area, we could use only a small number of such samples. Thus, our experiments have a higher error margin than is possible to achieve. However, our experience demonstrates strong potential for this approach for large and dense sensor networks. We are in the

process of acquiring a larger testbed that will make such large-scale experiments feasible.

Experimental research using distributed networks of smart sensors is in its infancy. While research groups are working on algorithmic and performance aspects of collaborative signal processing in the target tracking arena [4, 13], experimental work focusing on localized communication has not yet appeared in mainstream literature in our knowledge. We expect that our experience will be useful to researchers pursuing research in this direction.

References

[1] ATMEL 8-bit RISC Processor. http://www.atmel.com/products/prod23.htm.

[2] Crossbow Technology, Inc. http://www.crossbow.com.

[3] RF Monolithics. http://www.rfm.com/products/data/tr1000.pdf.

[4] R. Brooks, C. Griffin, and D. S. Friedlander. Self-organized distributed sensor network entity tracking. *International Journal of High Performance Computing Applications*, 2002. Special Issue on Sensor Networks; to appear.

[5] Nirupama Bulusu, John Heidemann, and Deborah Estrin. GPS-less low cost outdoor localization for very small devices. *IEEE Personal Communications, Special Issue on "Smart Spaces and Environments"*, 7(5):28–34, 2000.

[6] W.S. Conner, L. Krishnamurhty, and R. Want. Making everyday life easier using dense sensor networks. In *International Conference on Ubiquitous Computing (UbiComp '01)*, pages 49–55, October 2001.

[7] Deborah Estrin, Lewis Girod, and Greg Pottieand Mani Srivastava. Instrumenting the world with wireless sensor networks. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Salt Lake City, May 2001.

[8] Deborah Estrin, Ramesh Govindan, John Heidemann, and Satish Kumar. Next century challenges: Scalable coordination in sensor networks. In *Proceedings of the Fifth Annual International Conference on Mobile Computing and Networks (MobiCom)*, 1999.

[9] A. Mainwaring et. al. Wireless sensor networks for habitat monitoring. In *Proceedings of the First ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, pages 88–97, 2002.

[10] Jeffrey Hightower and Gaetano Borriello. Location systems for ubiquitous computing. *IEEE Computer Magazine*, 34(8):57–66, 2001.

[11] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D.E. Culler, and K.S.J. Pister. System architecture directions for networked sensors. In *Architectural Support for Programming Languages and Operating Systems*, pages 93–104, November 2000.

[12] J. Caffery Jr. A new approach to the geometry of toa location. In *IEEE Vehicular Technology Conference (VTC)*, pages 1943–1949, September 2000.

[13] Dan Li, Kerry Wong, Yu Hen Hu, and Akbar Sayeed. Detection, classification and tracking of targets in distributed sensor networks. *IEEE Signal Processing Magazine*, 19(2), 2002.

[14] Q. Li, M. DeRosa R. Peterson, and D. Rus. Reactive behavior in self-configuring sensor networks. In *CDROM Proceedings of the Eighth Annual International Conference on Mobile Computing and Networks (MobiCom)*, pages CD–15–CD–16, 2002. Poster paper.

[15] Thorsten von Eicken, David E. Culler, Seth Copen Goldstein, and Klaus Erik Schauser. Active messages: A mechanism for integrated communication and computation. In *19th International Symposium on Computer Architecture*, pages 256–266, Gold Coast, Australia, 1992.

[16] K. Whitehouse and D. Culler. Calibration as parameter estimation in sensor networks. In *Proceedings of the First ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, pages 59–67, 2002.

[17] A. Woo and D.E. Culler. A transmission control scheme for media access in sensor networks. In *Mobile Computing and Networking*, pages 221–235, 2001.