# Cooperative Tracking with Binary-Detection Sensor Networks

Kirill Mechitov Sameer Sundresh Youngmin Kwon
Gul Agha
University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA
{mechitov,sundresh,ykwon4,agha}@cs.uiuc.edu

**Abstract**

We present a novel method for tracking the movement of people or vehicles in open outdoor environments using sensor networks. Unlike other sensor network-based methods, which depend on determining distance to the target or the angle of arrival of the signal, our cooperative tracking approach requires only that a sensor be able to determine if an object is somewhere within the maximum detection range of the sensor. We propose *cooperative tracking* as a method for tracking moving objects and extrapolating their paths in the short term. By combining data from neighboring sensors, this approach enables tracking with a resolution higher than that of the individual sensors being used. We employ statistical estimation and approximation techniques to further increase the tracking precision, and to enable the system to exploit the tradeoff between accuracy and timeliness of the results. We analyze the behavior of the cooperative tracking algorithm through simulation, focusing on the effects of approximation techniques on the quality of estimates achieved. This work focuses on acoustic tracking, however the presented methodology is applicable to any sensing modality where the sensing range is relatively uniform.

## 1 Introduction

Recent developments in wireless network technology and the advent of small, inexpensive microprocessors have led to the emergence of *networked embedded systems* as a new kind of computing platform. By joining a multitude of embedded processors in a network, this platform enables the development of novel applications. Wireless sensor networks [11] are networked embedded systems in which nodes are equipped with sensors. The idea of using a network of "smart" sensors is attractive for a number of reasons:

- embedded processors are relatively cheap and have a small physical size

- networks of smart sensors may potentially scale to many thousands of nodes and thus cover a wide geographical area
- the networks may tolerate failures of individual sensor nodes
- because they use wireless communication and self-organizing ad hoc networking, deploying wireless sensor networks may be as easy as placing the nodes in the area of interest

Tracking, which involves identifying an object by its particular sensor signature and determining its path over a period of time, is one of the applications that can benefit from exploiting these characteristics of sensor networks. The inherent parallelism of distributed sensors makes it possible to track multiple objects simultaneously, while the relatively low cost and ease of deployment enable the use of sensor network-based tracking systems in remote or inaccessible locations, and when they need to be deployed on short notice. Situations where such conditions occur, and sensor network-based tracking systems can be most beneficial, include tracking rescue workers in emergency operations, military targets for reconnaissance, and monitoring traffic in future transportation systems such as intelligent highways [4].

On the other hand, there are a number of difficulties in using networks of embedded processors. The sensor nodes are subject to strong resource constraints, such as slow processors, small memory sizes, short battery life and a low-bandwidth, unreliable network connection. The requirement of low power consumption places additional limits on the use of these resources. Combined with timeliness requirements for most sensor data and a dynamic network topology, these constraints create an environment unsuitable for many traditional distributed algorithms. Algorithms for wireless sensor networks must have low communication overhead, rely as much as possible on local information, adapt to failures and changes in network conditions, and produce results in a timely fashion.

We propose *cooperative tracking* as a solution for tracking objects using sensor networks, and show that cooperative tracking may achieve a high degree of precision while meeting the above-mentioned constraints of networked embedded systems. Our approach uses distributed sensing to identify an object and determine its approximate position, and local coordination and processing of sensor data to further refine the position estimate. The salient characteristics of the cooperative tracking approach are that it achieves resolution that is finer than that of the individual sensors being used and that it provides early estimates of the object's position and velocity. Thus cooperative tracking is useful for short-term extrapolation of the object's path.

This paper considers an acoustic tracking system for wireless sensor networks as a practical application of the cooperative tracking methodology. Acoustic tracking relies on a network of microphone-equipped sensor nodes to track an object by its characteristic "acoustic signature." We provide results of simulations that show the high degree of precision and scalability achieved by the

algorithm. We also present a small-scale prototype implementation to demonstrate feasibility of the approach and validate the assumptions of the simulation.

The remainder of this paper is organized as follows. We describe the cooperative tracking problem and our approach to solving it in Section 2. In Section 3 we discuss our simulation results, while Section 4 talks about the prototype implementation of the cooperative acoustic tracking system and presents some experimental results. Section 5 presents an overview of related work, and Section 6 concludes the paper.

# 2 Cooperative Acoustic Tracking

We define the tracking problem in the context of sensor networks by specifying a model for the sensor network and for the objects being tracked. We then present our cooperative tracking algorithm for acoustic sensing.

## 2.1 Model

In the real world, objects can move arbitrarily, *i.e.* possibly changing speed and direction at any time. The representation of such arbitrary paths may be cumbersome, and unnecessarily complex for the purpose of tracking the object's path with a reasonable degree of precision. Instead, an approximation of the path can be considered. We use *piecewise linear approximation* to represent the path of the tracked object. Although the object itself may move arbitrarily, we consider its path as a sequence of line segments along which the object moves with a constant speed. The degree to which the actual path diverges from its representation depends on several factors, including speed and turning radius of the object itself. For vehicles such as cars driving along highways the difference is quite small, whereas for a person walking a curved route with tight turns it may be significant. In either case, accuracy can be improved by increasing the resolution of the sensor network, either through increasing sensor density or by other means.

Thus the model of the sensor network is as important for tracking as is the model of the tracked object. A sensor network consists of a number of sensor nodes placed in the area of interest. We assume that the sensors are randomly placed in the environment with uniform distribution. Each node is equipped with a sensor (in the case of acoustic tracking, a microphone) and a radio for communication with nearby nodes. Since these embedded systems are designed to be small and cheap, the sensors they are equipped with are unlikely to be very sophisticated. Traditionally, tracking relies on sensors that are long range and can detect the direction of an object and the distance to it. This is not the case with sensor networks: the microphones used for acoustic tracking are likely to be short range, non-directional and poorly suited for detecting the distance to the sound source. For the purposes of this paper, we assume that only binary (on-off) detection can be used. It is possible to generalize this analysis if multi-level detection is feasible. Moreover, without proper calibration the detection

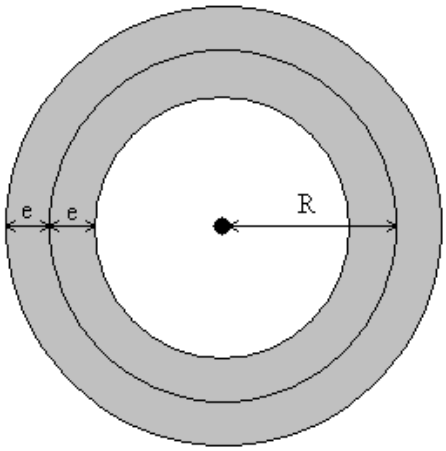


Figure 1: Model of a sensor. For nominal sensing range $R$, the object is always detected when it is $R - e$ away or closer, never detected beyond $R + e$, and has a non-negative chance of detection between $R - e$ and $R + e$.

range may be neither uniform nor exact. Figure 1 shows the model of a sensor considered in this paper. Given a sensor with a nominal (non-calibrated) range $R$, the object will always be detected if it is distance $R-e$ or less away from the sensor, detected some of the time between $R - e$ and $R + e$, and never detected beyond that range. We found that setting $e = 0.1R$ comes fairly close to the actual behavior of the sensors used in our experiments.

To track an object, it must be identified and its presence detected. For acoustic tracking, objects are identified based on their acoustic signature, which is a characteristic sound pattern or a set of frequencies unique to that object. For simplicity, we assume the object emits sound of a frequency not present in the environment, so there are no false positives. However, our results are fairly robust with respect to intermittent detection (false negatives) during the period of observations.

It is worth noting that the sensor model is generic enough to encompass other sensing modalities beyond acoustic. All that is required is a sensor with a relatively uniform range, as defined above, which is capable of differentiating the target from the environment. Magnetometer, a device that detects changes in magnetic fields, is one such sensor.

## 2.2 Algorithm

The simplest distributed tracking algorithm entails simply recording the times when each sensor detects the object and then performing line fitting on the resulting set of points. While simple, this approach is not very precise: it can only track the object with a resolution of the sensor range $R$. Moreover, if

a sensor detects the object more than once as it moves through the sensor's detection range, that information is lost.

The position of a stationary object, or a moving object for that matter, which is determined using this method is not very precise and depends heavily on the number, the detection range and precision of sensors that detect the sound. Instead of looking at a single position measurement, we are interested in the path of a moving object, which is a sequence of positions over a period of time. Combining a large number of somewhat imprecise position estimates distributed over space and time may yield surprisingly accurate results. Cooperative tracking addresses the problem of high-resolution tracking using sensor networks. It improves accuracy by combining information from neighboring sensors. The only requirement for cooperative tracking to be used is that the density of sensor nodes must be high enough for the sensing ranges of several sensors to overlap. When the object of interest enters the region where multiple sensors can detect it, its position can be pinned down with a higher degree of accuracy, since the intersection area is smaller than the detection area of a single node. The outline of a generic cooperative tracking algorithm is as follows:

1. Each node records the duration for which the object is in its range.
2. Neighboring nodes exchange these times and their locations.
3. For each point in time, the object's estimated position is computed as a weighted average of the detecting nodes' locations.
4. A line fitting algorithm is run on the resulting set of points.

Several of these steps require careful consideration. First, the algorithm implicitly assumes that the nodes' clocks are synchronized, and that the nodes know their locations. Discussion of time synchronization and localization in sensor networks is beyond the scope of this paper, however we give references to related work on these problems in Section 5. Second, we obtain a position reading by a weighted average of the locations of the nodes that detected the sound at a given instant, but the exact weighting scheme is not specified. This is an important issue, as selecting an appropriate scheme will improve accuracy, while a poor choice might be detrimental to it.

The simplest choice is to assign equal weights to all sensors' readings. This effectively puts the estimate of the object's position at the centroid of the polygon with sensors acting as vertices. This is a safe choice, and intuitively it should be more accurate than non-cooperative tracking. However it is possible to do even better. Consider Figure 2: sensors that are closer to the path of the target will stay in sensor range for a longer duration. Thus to increase accuracy, the weight of a sensor's reading should be proportional to some function of the duration for which the target has been in range of that sensor. We use simulation to evaluate several weighting schemes for cooperative tracking. Simulation results are presented in Section 3.

Once the individual position estimates are computed, the final step of the *line fitting* algorithm can begin. We use least squares regression to find the

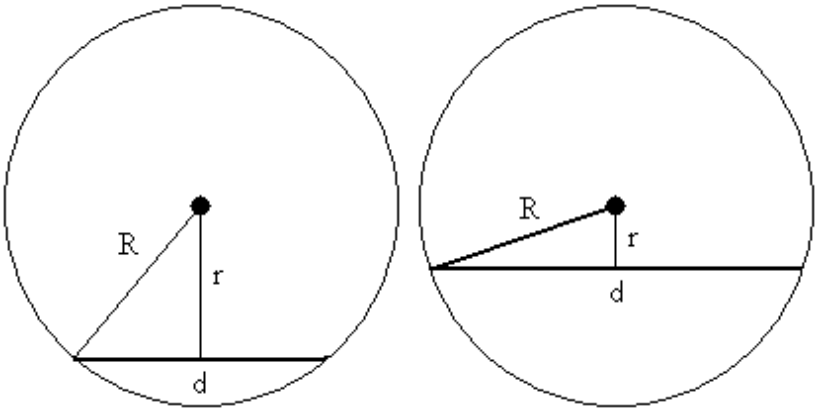


Figure 2: If the object's speed is constant, detection time is directly proportional to path segment $d$ and inversely proportional to distance $r$ from the sensor to object's trajectory.

equation of the line. It is interesting to note that the duration-based weighting scheme for position estimates moves the points closer to the actual path, thus reducing variance in the least squares computation. Also important is the fact that the multi-step approach enables early estimates of the path to be computed, so that continuous refinement is possible as more data points become available. The resulting equation of the line extrapolates the path of the object until it changes course sharply. This information may be used by the system, *e.g.* for asynchronous wakeup of nodes likely to be in its path.

## 2.3 Data Aggregation

The final step of the algorithm involves performing a line fitting computation on the set of all the position estimates (or some subset of them). Unlike position estimates, which can be performed in a distributed manner with only local communication, this necessitates collecting sensor readings from many sensor nodes at a centralized location for processing. This process is called data aggregation, and it is present in one form or another in virtually all sensor network applications. The main concerns for data aggregation are timeliness and resource usage. Timeliness, with respect to sensor data, is critical to real-time monitoring and control applications where stale data is useless or even detrimental. Resources, in particular network bandwidth and message buffers, are quite scarce in networked embedded systems. Low bandwidth of small wireless transmitters and the potential for contention with other messages drastically limit the amount of data that can pass through the network.

We assume that some nodes in the sensor network are *gateways* — nodes connected to outside networks such as the Internet. To process the data from

the sensor network, it needs to be sent through one of these gateway nodes to the more powerful computers connected to the outside network. To do this efficiently, we construct a tree rooted at each gateway and spanning the entire network. Each sensor node in the tree collects data from its children and sends it up the tree to either the closest or the least busy gateway. This scheme addresses the conflicting requirements of low bandwidth usage and timeliness of data: a near-shortest path is always taken, unless its links are overloaded. We assume that the outside network is low latency and high bandwidth, so it does not matter to which gateway the data is sent. More efficient data aggregation methods are a subject for future work.

# 3 Simulation

We implemented a simulator for a cooperative acoustic tracking system in order to examine the different weighting schemes for improving the accuracy of the estimates, as well as to evaluate the effects of parameters such as density on the accuracy of cooperative tracking. Likewise, we wanted to test whether the data aggregation method described in Section 2.3 scales well as the number of nodes increases. We simulate a 100-node sensor network with a single mobile object, both behaving according to the model of Section 2.1.

The object moves through the sensor network in a straight line at a constant speed of $1\,R/\mathrm{s}$. The network density is $1\,\mathrm{node}/R^2$, and the nominal range for all sensors is $R = 1.2\,\mathrm{m}$. The actual sensor range varies from $0.9\,R$ to $1.1\,R$ according to an abridged Gaussian distribution. Since we express the main parameters, object's speed and sensor density, in terms of the sensor radius $R$, the results can be generalized to tracking objects traveling at a different speed by increasing density accordingly.

## 3.1 Results

The first metric we consider is estimation error, measured a percentage difference between the measured and the actual slope, with 180° error equivalent to 100%. The main source of error in this case is the approximation of the object's position based on the positions of the sensors that detect it, thus different weighting schemes should have a direct effect on the magnitude of the error.

Figure 3 shows the results of the simulation with three different weighting schemes for the object's position estimates. The baseline method assigns equal weights to all sensors when computing the position estimate. This is equivalent to placing the estimate at the centroid of the polygon formed by the detecting sensors. It yields the least accurate results. The other two schemes exploit the fact that if the sensor lies close to the path of the object, the detection period will be longer.

The second scheme involves assigning sensor weights that are inversely proportional to the estimated perpendicular distance to the path of the object. The following formula is used to compute the weight for sensor $i$:

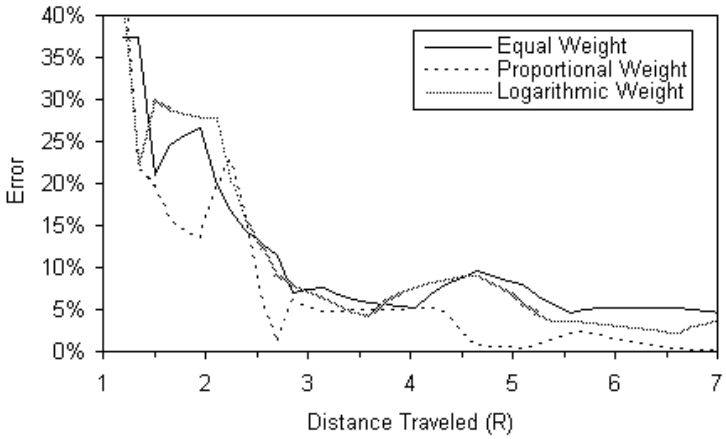


Figure 3: Estimation error under the three weighting schemes. Error is measured as a percentage difference between the measured and the actual slope (100% is 180°), while Distance Traveled is in units normalized to sensor range $R$.

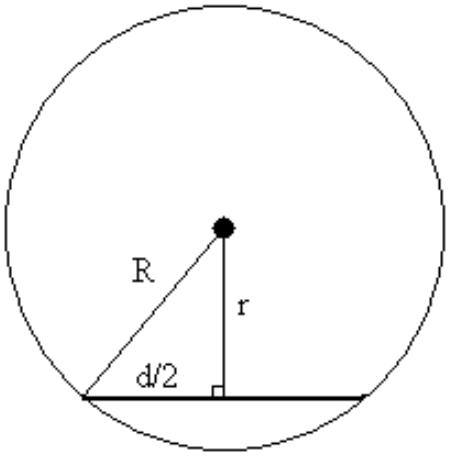


Figure 4: Weight is assigned to sensor readings based on the computation of $r$, the perpendicular distance from the sensor to the path of the tracked object.

$$w_i = \frac{1}{\sqrt{r^2 - 0.25(v * (t_i - 1/f))^2}}$$

Here $r$ is the sensor radius, $v$ is the estimated speed of the object, $t_i$ is the duration of object detection, and $f$ is the sensor sampling frequency. Somewhat unsightly, the formula is simply the geometrically-derived expression for the perpendicular distance from the sensor to the secant that corresponds to the estimated path (Figure 4). In the special case where this distance (the denominator) is zero, a very large positive weight is assigned to the sensor since it is known to be directly on the path of the object. This scheme outperforms the baseline case, with the magnitude of the error reduced 2-7 times after the position estimates have stabilized. It does, however, depend on the availability of an estimate of the tracked object's speed. The approximate speed can be computed using the same (time, location) tuples used for determining the path, although it is much less sensitive to error.

The third scheme is heuristic; it assigns weights to sensors using the formula $w_i = \ln(1+t_i)$, where $t_i$ is the duration for which the sensor has heard the object. This weight, derived empirically, biases the least squares regression to favor the nodes that are closer to the actual path of the object. This reduces the effect of errors and imprecision of sensors far from the path. Our intuition is supported by the results, as the error is consistently less than in the unweighted scheme after the readings stabilize. Based on the same principle as the proportional weight method, the logarithmic weight method does not rely on estimating the speed of the object. This reduces the amount of inter-node communication needed to come up with a position estimate; however it cannot match the performance of the proportional weight scheme.

## 3.2 Discussion

Despite the good performance, the proportional weight scheme is still pessimistic in one sense. When a sensor first detects a signal, it is given low weight because it assumes that the path through its sensor range is along a short secant, even if the path lies directly along the diameter. This problem can be partially addressed by retroactively increasing the weights on previous readings once the object has been in range for a longer duration. This can improve the final estimate, but this method still has the problem of assigning too much weight to old readings and too little weight to new ones. This is especially significant for piecewise linear path estimation, where older readings, before the object made a sharp turn, may be much less accurate than the new ones.

Estimation accuracy can be improved by exploiting the earlier estimates of the path to more accurately predict the distance that the object will travel through the field of view of the sensor, and thus how much weight is assigned to these readings. If the estimates are fairly accurate, the path estimates will converge faster. On the other hand, if the estimates are inaccurate the quality of further estimates may be degraded. If we can assign confidence values to intermediate estimates, these values can be used to determine if using the estimate over the original weighting scheme is beneficial.

Another approach that may decrease estimation error involves weighting sensor readings proportionally to the size of area in which the object may be located.

This potential area is the intersection of the sensing fields of all sensors that detect a object, thus more simultaneous readings from geographically-distributed sensors result in a smaller area and thus better accuracy. The computation of potential areas is significantly more CPU-intensive than the current weighting scheme and may be inappropriate for sensor networks, but computation could be done off-line or on-line on a more powerful computer connected to the network. Determining whether this is feasible or worthwhile is part of our future work.

# 4 Experiments

We now describe the implementation of a prototype cooperative acoustic tracking system. The goal is to demonstrate the feasibility of the approach through experimental data, and to act as a reality check for the simulations. We first describe the system and then present the results of several experiments conducted on it.

## 4.1 Sensor Hardware

We implemented the acoustic tracking system on a network of Mica motes. A mote (Figure 5) is a prototype networked sensor platform. Its principal hardware characteristics include an 8-bit Atmel processor running at 4 MHz, 8 KB of program memory, 512 bytes of RAM and a single-channel 916 MHz RF transceiver capable of providing 19.2 Kbps of raw bandwidth. Two AA batteries act as the power source. A sensor board featuring a variety of sensors, as well as a microphone and a beeper, can be attached to the mote. The power (and hence the range) of the RF transceiver and the microphone can be controlled by the application via a potentiometer setting. The TinyOS 0.6.1 operating system [3] provides the application programmer interface to the hardware. A small, event-driven operating system developed specifically for networked sensors, TinyOS is lightweight but sufficiently powerful to provide a foundation for building intelligent sensor applications.

## 4.2 Cooperative Acoustic Tracking

The wireless sensor network for acoustic tracking is made up of 16 motes, laid out in a regular $4 \times 4$ grid. The sensor nodes are placed 0.3m apart, and the reliable detection range of the microphone with the given gain setting is $R = 0.5\,\mathrm{m}$. To simulate a localization service, each mote is preprogrammed with its physical coordinates. Another mote is used as the mobile target. The beeper on this mote's sensor board is set to emit a constant tone detectable by the sensor grid. The target mote moves linearly through the wireless sensor grid at a constant speed of $0.2\,\mathrm{m/s} = 0.4\,R/\mathrm{s}$. It should be noted that the RF signal range, microphone power and the speed of the target are significantly reduced for the purposes of the laboratory experiment. The same setup can cover a wider

Figure 5: The Mica mote processor board. A sensor board featuring a microphone and a variety of other sensors can be attached, turning the mote into a wireless sensor platform.

area and track faster moving targets. We use the logarithmic weighting scheme for computing position estimates, since it is more efficient than the unweighted case, yet does not require an estimate of the velocity of the object *a priori*.

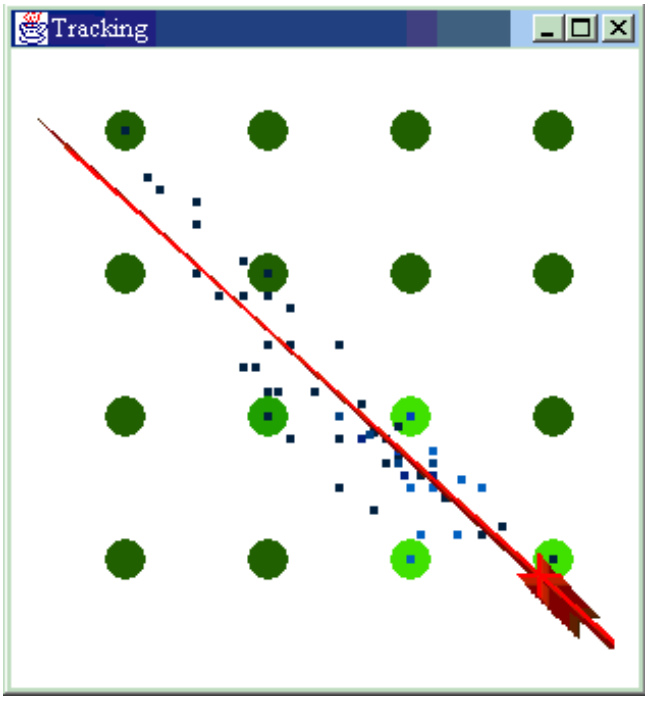


Figure 6: Tracking display. Circles represent sensors, dots are position estimates, the line is the object's estimated path, and "× marks the spot" of the object's current position.

## 4.3 Results

Experiments on the acoustic tracking system described above were conducted to study the accuracy of our cooperative tracking algorithm under real-world conditions and to assess the effectiveness of the data aggregation scheme at reducing congestion in the network.

**Tracking Accuracy**

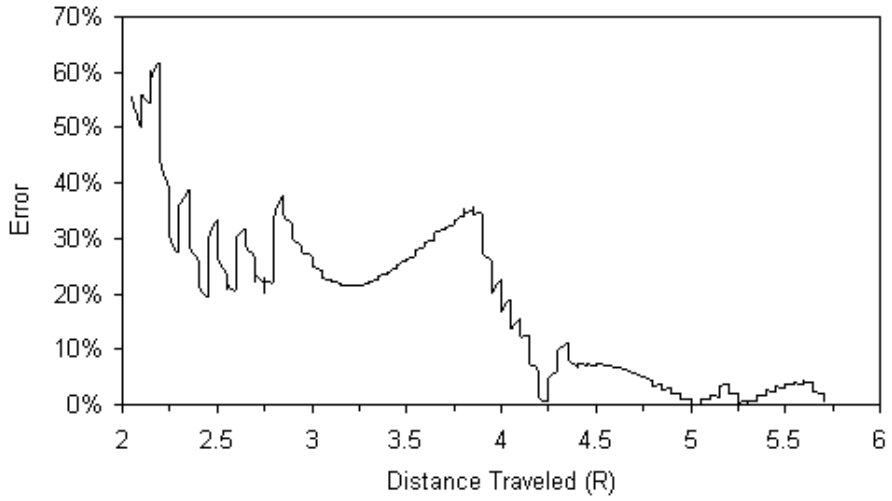


Figure 7: Accuracy of cooperative acoustic tracking. Error is measured as the percentage difference between the measured and the actual slope (100% is 180°), while Distance Traveled is in units normalized to sensor range $R$.

The principal metrics for the performance of a tracking application are how close the computed path is to the measured phenomenon, and how quickly the result becomes available. The angle between the computed and the measured slopes is again taken as the measure of quality of approximation. Figure 7 presents results of this experiment. Several characteristics of the graph are worth noting. First, it is encouraging that even with an imprecise target positioning method and a very limited number of sensors (at most 4 in a straight line) the magnitude of the error falls below 5% within seconds of initial detection. While the error is still greater than simulation results show, the difference is relatively small and may be due to the smaller scale of the experiment.

Also, the general shape of the curve is consistent with the simulation results. The two most significant differences are time — in simulation we did not consider warm-up time and network propagation delays — and sharp changes in the magnitude of the error, which can be attributed in part to the small number of

sensors and their arrangement on a grid during the experiment. These results confirm that the model used in the simulation, both for the moving object and the sensors, is consistent with reality. This is significant as an indicator that the relatively simple model of the sensor's detection field can be used in analysis and simulations in the future.

The speed with which accurate tracking is achieved is also significant. Although it is greatly affected by factors such as movement speed and inter-sensor separation distance, which were not varied in this experiment, it is still an important metric to consider when evaluating the usefulness of a tracking application. For example, the data collected 2-3 seconds after initial detection can be used to reduce network power consumption by powering down nodes that are far from the path of the object (the first few seconds are taken up by the setup phase, during which time no data is collected). Another feature of this algorithm is demonstrated by the gradual increase in accuracy as more and more data is collected and processed. This confirms our hypothesis that the accuracy of estimation can be directly controlled by varying the amount of data collected. This is an important tradeoff for designing time-critical systems that may need to sacrifice accuracy for timeliness.

**Network Congestion Control**

We also considered the impact of several techniques used to reduce congestion and increase throughput of the multi-hop wireless network. By far the most effective method of reducing interference caused by collisions, which is a significant cause of packet loss and corruption, is controlling the amount of data being sent out. By aggregating data and transmitting less frequently we were able to reduce packet loss in the network by 23%. Besides the dramatic reduction in packet loss, reducing the number of messages transmitted also reduces power consumption. We have not explicitly examined power consumption in this study, but prior research indicates that an active RF transmitter is a major source of power drain [13]. Although the effect of reducing the number of messages is obviously application-specific, these numbers emphasize the importance of careful application design with respect to the amount of data being exchanged and of the data aggregation methods employed.

# 5 Related Work

We base our assumptions about the capabilities of networked sensors on the Mica mote hardware and the TinyOS software [3]. This customizable open platform enables development and testing of sensor networks applications in the real-world conditions. One of the drawbacks of the development version of TinyOS used in our implementation is a poor MAC layer protocol, which leads to many dropped packets due to collisions. S-MAC, an alternate implementation, promises better performance as well as reduced power consumption [13]. Multi-hop routing is another service used implicitly by our tracking system.

Several routing algorithms for ad hoc wireless networks have been developed, *e.g.* DSDV [10] and AODV [9]. In the case of sensor networks, the nodes' knowledge of their locations may also be used to assist with routing, which tends to reduce overhead [5]. The problem of determining these locations without prior knowledge is addressed in [1, 12, 8]. Localization, as this process is called, is currently an active research area. Time synchronization is another service needed by cooperative tracking. While time synchronization research in traditional distributed systems focused to a large extent on fault tolerance (*e.g* [6]), recent research addressed the problems of precision and performance in time synchronization specifically for sensor networks [2]. Finally, data aggregation is an important component of many sensor network applications, including cooperative tracking. A simple but efficient data aggregation scheme was presented in Section 2.3. A more general and more powerful aggregation service for sensor networks is described in [7].

# 6 Conclusion

We developed the cooperative tracking method for using sensor networks to track mobile objects. This method achieves resolution finer than that of the sensors being used due to cooperative sensor data processing and a heuristic position estimation algorithm. Continuously refining path estimates as new data comes in, cooperative tracking also makes early results available very quickly for time-critical applications, while constantly improving the accuracy of the path estimates. A prototype implementation of the cooperative acoustic tracking system confirms the feasibility of this approach.

A lot of work remains for the refinement of the cooperative tracking algorithm. Heuristic position estimation can be further improved by considering the area where the node is among the criteria for assigning weight to sensor readings, and a deeper analysis of the measurement uncertainties needs to be performed. Perhaps even more importantly, there is a lot of room for improvement in sensor network infrastructure: services like data aggregation, routing and localization. As these services mature, cooperative tracking and other sensor network applications will be able to better utilize the available resources, further improving the quality of the results.

# 7 Acknowledgments

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Award No. F33615-01-C-1907. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of DARPA. The authors would like to thank Wooyoung Kim, Predrag Tosic, and the members of the OSL NEST group for the discussions and suggestions that helped improve this paper.

# References


[1] Nirupama Bulusu, John Heidemann, Deborah Estrin, and Tommy Tran. Self-configuring localization systems: Design and experimental evaluation. *ACM Transactions on Embedded Computing Systems*, page To appear., May 2003.

[2] Jeremy Elson, Lewis Girod, and Deborah Estrin. Fine-grained network time synchronization using reference broadcasts. In *Symposium on Operating System Design and Implementation*, December 2002.

[3] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, and Kristofer Pister. System architecture directions for network sensors. In *Architectural Support for Programming Languages and Operating Systems*, pages 93–104, 2000.

[4] R. Horowitz and P. Varaiya. Control design of an automated highway system. In *IEEE*, volume 88, pages 913–925, 2000.

[5] Y.-B. Ko and N. H. Vaidya. Location-aided routing (LAR) in mobile ad hoc networks. In *ACM/IEEE Int. Conf. on Mobile Computing and Networking*, October 1998.

[6] Leslie Lamport and P. M. Melliar-Smith. Synchronizing clocks in the presence of faults. *Journal of the Association for Computing Machinery*, 32(1):52–78, January 1985.

[7] Samuel Madden, Michael Franklin, Joseph Hellerstein, and Wei Hong. TAG: A tiny aggregation service for ad-hoc sensor networks. In *Symposium on Operating Systems Design and Implementation*, December 2002.

[8] Dragos Niculescu and Badri Nath. Ad hoc positioning system (APS). In *GLOBECOM*, November 2001.

[9] Charles E. Perkins, Elizabeth M. Belding-Royer, and Samir Das. Ad hoc on demand distance vector (AODV) routing. Technical report, IETF Internet Draft, march 2002.

[10] Charles E. Perkins and Pravin Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. *ACM SIGCOMM Computer Communication Review*, 24(4):234–244, October 1994.

[11] G. J. Pottie and W. J. Kaiser. Wireless integrated network sensors. *Communications of the ACM*, 45(5):51–58, May 2000.

[12] Andreas Savvides, Chih-Chieh Han, and Mani B. Strivastava. Dynamic fine-grained localization in ad-hoc networks of sensors. In *ACM/IEEE Int. Conf. on Mobile Computing and Networking*, pages 166–179. ACM Press, 2001.

[13] Alec Woo and David Culler. A transmission control scheme for media access in sensor networks. In *ACM/IEEE Int. Conf. on Mobile Computing and Networking*, 2001.