Acoustic Target Tracking Using Tiny Wireless Sensor Devices

Qixin Wang, Wei-Peng Chen, Rong Zheng, Kihwal Lee, and Lui Sha

Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801
{qwang4, wchen3, zheng4, klee7, lrs}@cs.uiuc.edu

Abstract. With the advancement of MEMS technologies, wireless networks consist of tiny sensor devices hold the promise of revolutionizing sensing in a wide range of application domains because of their flexibility, low cost and ease of deployment. However, the constrained computation power, battery power, storage capacity and communication bandwidth of the tiny devices pose challenging problems in the design and deployment of such systems. Target localization using acoustic signal with tiny wireless devices is a particularly difficult task due to the amount of signal processing and computation involved. In this paper, we provide an in-depth study of designing such wireless sensor networks for real-world acoustic tracking applications. We layout a cluster-based architecture to address the limitations of the tiny sensing devices. To achieve effective utilization of the scarce wireless bandwidth, a quality-driven paradigm to suppress redundant information and resolve contention is proposed. One instance of the quality-driven approach is implemented in the acoustic tracking system, where the quality of the tracking reports can be quantified numerically. We demonstrate the effectiveness of our proposed architecture and protocols using a sensor network testbed based on UCBerkeley mica motes. Considering the performance limitations of tiny sensor devices, the achieved acoustic target tracking accuracy is extraordinarily good. Our experimental study also shows that the acoustic target tracking quality can be indeed measured and used to assist resource allocation decisions. This application-driven design and implementation exercises also serve to identify important areas of further work in in-network processing and communications.

Index Terms — wireless sensor network, sink tree routing, acoustic target tracking, quality driven, redundancy suppression.

1 Introduction

Fueled by the increasing capabilities and ever-declining cost of computing and communication devices, wireless networks consisting of a large number of tiny

sensors hold the promise of revolutionizing a wide range of application domains such as battlefield surveillance, machine failure diagnosis, biological detection, home security, smart spaces, inventory tracking etc[8]. In surveillance applications, visual and audio data are two key sources of information. Airborne/satellite cameras can easily monitor a wide area while the collection of acoustic information can only be done close to the source due to the limited distance of propagation of sound in the air. As a result, we currently have many eyes in the sky but not enough ears on the ground, so to speak. This makes acoustic tracking of mobile targets using tiny sensing devices very attractive, as massive deployment of wireless sensors in large area can provide more accurate and timely information about the geographical location of the targets. The basic idea of acoustic tracking is to detect the location of a target by analyzing the specific cues such as delay and amplitude received by multiple sensors.

Generally speaking, wireless sensor networks consist of tiny sensory devices deployed in a region of interest. Each device has limited processing and wireless communication capabilities, which enable it to gather information from the environment and deliver this information to actuators for appropriate actions. The major challenges in design and deployment of such wireless sensor networks are the constrained computation power, battery power, storage capacity and communication bandwidth of the tiny devices. Target tracking using acoustic signal is a particular daunting task for the following reasons,

- Acoustic tracking needs collaborative communication/computation among multiple sensors. The information gathered by a single sensor is usually incomplete and inaccurate.
- Acoustic tracking requires a significant amount of signal processing and computation to detect and locate the sources of interest.
- The reports generated by the sensing components need to be delivered to the actuators in a timely fashion. Out-dated reports are of little use.

In this paper, we provide an in-depth study of the *architecture and algorithmic issues* in applying networks of *tiny* wireless sensors to real-world acoustic tracking applications. *The acoustic signal processing techniques used can be quite simplistic, nevertheless, it provides a context for our research.* In addition, in this paper, we only deal with tracking *impulsive* acoustic signals, such as foot steps, sniper shots etc. However, the networking aspect of the system is applicable to any type of target localization.

The overall system architecture consists of two self-contained components, the acoustic target tracking subsystem, which deals with the detection, processing and triangulation of impulsive acoustic signals; and the communication subsystem, which is responsible for reporting high quality tracking results to the data sink in a timely fashion. To address the limited computational and battery power of wireless sensor devices, we organize sensors into clusters. Sensors in each cluster coordinate in sensing and communication to perform the sensing task. To deal with the inaccuracy in measurement and unreliability typical with low-end device in remote or hostile environment, we explore the redundancy in

a large number of sensors to obtain more robust results. To achieve effective utilization of the scarce wireless bandwidth, a quality-driven scheme is proposed to suppress redundant data and resolve channel contention. The novelty of quality-driven scheme is that it aims to increase the flow of information as compared to raw bits. One instance of the quality-driven approach is implemented in our testbed, where the quality of the acoustic tracking reports can be quantified.

We demonstrate the effectiveness of our proposed architecture and protocols using a sensor network testbed consisting a number of UCBerkeley mica mote sensor nodes and a few pc/104 single board computers. In our experiments, with an acoustic sensor density of $0.125/ft^2$, we can locate a sound source with an average error of 13.8 inches, among which 35% of the errors are less than 3 inches and 48.3% of the errors are less than 6 inches. Our experimental study shows that the data quality can be indeed measured and used to assist resource allocation decisions. This application-oriented design and implementation exercises also serve to identify and provide insights to important areas in in-network processing and communications.

The organization of this paper is as follows. After the introductory part, we give a brief overview of hardware constraints and the system architecture. Section 3 presents the design and implementation of time synchronization, acoustic signal processing and triangulation for acoustic target tracking. In Section 4, a quality-driven in-network signal processing and communication scheme for redundancy suppression and contention resolution is proposed, together with a novel multi-parent sink-tree routing algorithm. In Section 5, we evaluate the proposed architecture and algorithms using a sensor network testbed and conclude the paper in Section 6 with a list of future work.

2 System Overview

2.1 Networked Wireless Sensor Device

As stated in Section 1, our mission is to implement acoustic target tracking using networked sensors. In designing our system, we face the following challenges:

1. Limited hardware capability (in terms of CPU MIPS, sampling rate, program and data memory, wireless bandwidth etc.);
2. Error- and failure-proneness (e.g., due to the low sampling rate and low sampling accuracy, loss of interrupt events, power constraints, physical damage etc.);
3. Difficulty in programming and debugging embedded systems.

The specific hardware we use is *mica mote*, developed by UC Berkeley [5] [14]. Mica mote uses the ATmega103L micro-controller [3] with a 4MHz CPU cycle frequency and a 4KB data RAM. The wireless networking hardware component on the mica motes is the RFM TR1000 radio transceiver [7], which operates at the unique radio frequency of 916.5MHz. It can achieve a maximal application layer throughput of approximately 520Bytes/sec [13]. The acoustic sensor

module on mica mote can at most reach a stable sampling rate of 4kHz. A component-based OS – TinyOS [5] is used as the software system architecture for manipulating motes.

As will become clearer later on, the hardware capability is very limited for data-centric, computational-intensive acoustic tracking applications.

2.2 System Architecture

Due to the aforementioned hardware and software limitation of tiny devices, our system architecture design has to take extra care on the application requirement, availability, robustness and manageability. We adopt the following design philosophies, i) exploring the redundancy in large number of sensing devices, ii) divide-and-conquering acoustic tracking tasks by carrying out role differentiation, and iii) targeting for effective utilization of limited resources rather than nominal utilization. Redundancy is desirable not only for the purpose of better availability (to counteract the impact of high device failure rate), but also for the purpose of robustness. Redundant data can statistically mitigate the negative impact of errors. Role differentiation splits the complex nationalities of the entire system into sub-tasks that are affordable and manageable for each one of the tiny devices. Increasing *effective* resource utilization makes use of limited resources to better serve the application requirements.

Specifically, we divide the overall system into two subsystems: the *acoustic target tracking subsystem* and the *communication subsystem*.

Acoustic target tracking subsystem: The acoustic target tracking subsystem consists of multiple *sensory clusters*. A sensory cluster is the primary unit for tracking the locations of acoustic targets. It has a *cluster head* and several *slavery acoustic sensors* which jointly monitor a specific area. A cluster can be formed using mechanisms such as the one used in Jini [12] to account various application-specific factors (such as geology, algorithm, device capability etc.) Dynamic clustering for moving targets is one of our ongoing research work. Redundancy is achieved by deploying extra sensors within a cluster and allowing the monitoring areas of adjacent clusters to overlap with each other. Traditionally, acoustic tracking needs only three sensors to carry out the *triangulation*. However, in our implementation, we require more than three sensors in each cluster to obtain one tracking report to combat the inaccuracy of individual sensor's data. Role differentiation is done by assigning cluster head and sensors different jobs and coordinating them to jointly carry out acoustic target tracking.

We assume that geographical location of each slavery sensor can be obtained via mechanisms such as described [4, 10]. Over the runtime of the system, the cluster head and all its slavery sensors are clock synchronized. When a sound with the specified signature arrives, all sensors record the sound and timing information and report them to the cluster head. By analyzing the differences of the sound arrival times among slavery sensors, the cluster head can estimate the location of the sound source using triangulation and then report the tracking result back to the *data sink* via the *communication subsystem*.

The cluster head needs to handle a significant amount of computation, which is beyond the capability of the current generation of mica motes. Therefore, in our implementation, we use pc/104 single board computers [6] as cluster heads, which are widely used in embedded systems.

Communication Subsystem: The communication subsystem is responsible for relaying the tracking reports from cluster heads to the *data sink*. In order to achieve high *effective* throughput and low latency, we propose a novel scheme called *quality-driven* redundancy suppression and contention resolution to speed up the delivery of higher quality tracking reports and suppress inferior/erroneous reports. Note, the redundancy of acoustic sensory motes reduces the error in a single cluster and thus is desirable. However, due to the overlapping of clusters, multiple reports can be generated for a single acoustic event across multiple adjacent clusters, which may unnecessarily lead to congestion of the network or waste of network resources. Quality-driven redundancy suppression picks the best one from the multiple tracking reports, thus accomplishes the goal of achieving better robustness out of redundant tracking reports. More importantly, quality-driven redundancy suppression and contention resolution is the major mechanism that realized the idea of *effective* utilization of communication subsystem rather than *nominal* utilization. To relay the report to the data sink we introduce multi-parent sink tree routing scheme to provide fast local recovery and higher message delivery ratio.

3 Acoustic Target Tracking Subsystem

In this section, we present the detailed design of the key components for the acoustic tracking system including the time synchronization, onset detection, cross-correlation, triangulation, and determination of the quality of results.

3.1 Time Synchronization

We use delay-based triangulation to locate impulsive sound sources, hence accurate timing information is a necessity. Specifically, all the sensors within the same cluster have to be time-synchronized. The method of time synchronization we adopt in the system is *Reference-Broadcast Synchronization* (RBS)[2], which is a light-weight scheme that can achieve high accuracy. In RBS, a head node broadcasts reference radio beacons to its neighbors. Each receiver records the arrival time based on its local clock and sends this information back to the originator of the head node. Under the assumption that the broadcasted radio beacon arrives at all receivers simultaneously, after a few rounds of the beaconing and replies, the head node can obtain mapping functions of clock readings between any pair of receivers, using statistically methods such as least square linear regression. To this end, the head node can convert any receiver's clock readings into a universal clock reading.

In our implementation, we have a initialization phase when n ($n = 12$ in our practice) rounds of beaconing and replies are performed. After the initialization, timing information can be piggybacked on the packets exchanged between the cluster head and the sensors. Therefore, the clocks can be calibrated without introducing extra control overheads. Experiment results show that the distortion of clock readings can be kept within $30\mu s$, which is sufficient for our delay-based acoustic tracking.

3.2 Onset detection

Due to the limited computation capability, the current generation of *mica motes* are not capable of sampling and processing acoustic data concurrently. Instead, major functions such as sensing, wireless transceiving and processing have to be serialized. Moreover, because of the limited memory in *motes*, acoustic samples have to be stored in a circular buffer. In order to avoid buffer overflow for useful sample data , an onset detection mechanism is needed to instruct sensors to stop sampling data once the interested acoustic signal is captured. The way a sensor determines whether the incoming acoustic signal is of potential interest is based on the magnitude of the signal. A small sliding window is used to compute the moving average of the magnitude of signals. If the energy within the window exceeds a threshold, the sensor assumes that the current time is close to the onset point of the acoustic signal. The sensor continues recording data into the circular buffer until it winds back and reaches a prelude point prior to the onset point. Once the sound of interest is captured, post-processing is conducted separately at both the cluster head and the sensors. The cluster head extracts the sound signature from the recorded samples and broadcasts it to the sensors in its own cluster. Upon receiving the signature packet from the cluster head, sensors apply cross-correlation to compare the received signature with the buffered data.

3.3 Cross-Correlation

After receiving the sound signature from the cluster head, each sensor cross-correlates the received signature with buffered data to extract the desired pattern and determine the starting portion of signal. There are several advantages in choosing the starting portion as the reference portion. First, the starting portion is less susceptible to echoes. In in-door environments, the effect of echoes is quite significant. Fortunately, the echo is not presented in the starting portion of acoustic signal unless the sensor is very close to a wall. Second, the uniqueness of onset point and the salient change in sound wave shape at the onset point makes the starting portion very easy to be consistently located among distributed independent sensors.

The procedures of cross-correlation are summarized in Fig. 1. Two preprocessing procedures are applied to buffered data before cross-correlation. The first step is to remove the interference of noise. The average magnitude of noise is calculated first, and then the samples whose values are close to the average are replaced with the average so that the results of correlation do not ripple due

to the oscillation of noises. In the second step, the signal is passed through a second-order Butterworth low-pass filter to remove the high frequency components. After the pre-processing, cross-correlation is applied to the filtered data with the received sound signature to do pattern matching. The final step is to find the first significant peak in the correlation result using thresholding and thus the arrival time is extracted.

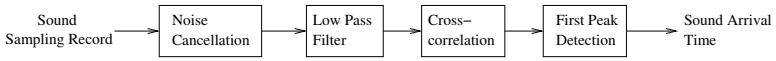


Fig. 1. *Procedures of cross-correlation for sensors*

3.4 Triangulation (Sound Source Locating) and Evaluation of *Quality Rank*

Upon receiving the *sound arrival times* from sensors in the cluster, the cluster head translates the time into the universal reference time before executing the triangulation.

Triangulation is done by comparing the differences in sound propagation delays from the source to different acoustic sensors. Suppose the location of sound source is (x, y) and the sound is generated at time t. If a sensor M_1 located at (x_1, y_1) detects the sound reaches it at time t_1 (i.e. sound arrival time). We shall have the following equation:

$$\sqrt{(x - x_1)^2 + (y - y_1)^2} = (t_1 - t) \cdot v \tag{1}$$

where v is the velocity of sound.

Theoretically triangulation can use three such equations generated by three sensors to analytically compute the sound source location. In practice, due to the coarse-grained acoustic sampling data of low-performance sensors, errors generated in cross-correlation are not negligible (for example, the errors can vary from 0.1 to 0.5ms). The above theoretical approach is mostly impractical. How to counteract this problem is a big research topic. Relevant works can be found in [11][1][9] etc. However, this topic is out of the scope of this paper. In this paper we simply consider a maximum likelihood (ML) based heuristic as a context for our research.

Algorithm Description: Consider n sensors, for each hypothetical source location $(\tilde{x}, \tilde{y})$ in the field A, there is a vector $\tilde{p}(\tilde{x}, \tilde{y}) = (\tilde{d_1}, \tilde{d_2}, \ldots, \tilde{d_n})$ representing the theoretical propagation delay to each one of the n sensors. Let $p = (d_1, d_2, \ldots, d_n)$ be the observed propagation delay vector based on the reports gathered at the cluster head for a single sound events. Then, the estimated location of the sound source $(\hat{x}, \hat{y})$ is given by

$$(\hat{x}, \hat{y}) = \arg_{(\tilde{x}, \tilde{y}) \in A} \min d(\tilde{p}(\tilde{x}, \tilde{y}), p) \tag{2}$$

Where d is an algorithm specific difference measurement scheme.

This algorithm only involves multiplicity, additions and comparisons. Compared to equation-based solution, which usually requires division and solving of linear and quadratic equations, our algorithm is more robust to floating point errors or degenerations.

However, since there are infinite points in area A, to make the algorithm work, the monitored area of a cluster is first divided into N-by-N grids (in our implementation, we choose N so that the granularity of the grid is $3 \times 3\text{inch}^2$). Therefore the time complexity of the algorithm is roughly $O(N^2)$. In addition, it is possible to throw out out-of-bound readings based on the physical laws to avoid unnecessarily degradation of the triangulation result.

The confidence level of estimated location can also be approximated by the percentage of $\tilde{p}$'s elements that falls within the ε boundary of p. This percentage is defined as the *quality rank* of the location report. The higher the percentage, the more confidence in the sensed result and thus the higher the quality rank. Determination of the quality rank of triangulation results is important. In the case of multiple clusters (whose monitored areas may overlap), multiple reports for the same sound event may be generated and delivered to the data sink. Ideally, only the cluster head with the best quality rank (or highest confidence) need to send the report back. Inferior quality reports should be suppressed to better utilize the limited bandwidth and computational power in the wireless networks. This point will be further illustrated in Section 4.1.

The pseudo code for locating the sound source and determining quality rank is given in Fig. 2, where, $\hat{\Delta}$ is the estimation of the fixed time difference between the time reference systems used by $\tilde{p}$ and p. Because $\tilde{p}$ uses the relative time which takes actual sound event happening time as 0; while p can only use universal time which takes the time synchronization initialization time as 0.

4 Communication Subsystem

Communication subsystem serves to forward the tracking report from cluster head to the *data sink*. We use the default MAC and routing protocol (i.e., CSMA and sink tree) of TinyOS [5] as a baseline. In order to achieve high robustness in acoustic tracking report, availability, effective throughput and low delay, we propose the idea of *Quality-driven redundancy suppression and contention resolution* (QDR) and *multi-parent routing* .

4.1 Quality-driven Redundancy Suppression and Contention Resolution

As mentioned before, in the acoustic target tracking subsystem, for the purpose of providing better robustness and availability, we allow the overlapping of clusters' monitoring areas to generate redundant reports for each sound event. Therefore, a mechanism is needed to quantify the quality of reports and select the one with the best quality. Inferior redundant reports can be suppressed from

1. **LocateSoundSource**(observed propagation delay vector $p = (d_1, \ldots, d_n)$):
2. mark apparently invalid d_i s in p
3. $m \leftarrow$ number of valid elements in p
4. **for each** point $(\tilde{x}, \tilde{y}) \in Grid$
5. calculate $\tilde{p} = (\tilde{d_1}, \cdots, \tilde{d_n})$ for $(\tilde{x}, \tilde{y})$ according to Equation (1)
6. **for each** valid $d_i \in p$
7. $\delta_i \leftarrow \tilde{d_i} - d_i$
8. $\hat{\Delta} \leftarrow$ average of δ_i s
9. $vote \leftarrow 0$
10. **for each** valid $d_i \in p$
11. $err_i \leftarrow \delta_i - \hat{\Delta}$
12. **if** $|err_i| \leq \varepsilon$ **then** $vote \leftarrow vote + 1$
13. $QualityRank_{(\tilde{x},\tilde{y})} \leftarrow \frac{vote}{m}$
14. $(\hat{x}, \hat{y}) \leftarrow$ the $(\tilde{x}, \tilde{y})$ with the highest $QualityRank$
15. $\hat{Q} \leftarrow (\hat{x}, \hat{y})$'s corresponding $QualityRank$
16. **return** $(\hat{x}, \hat{y})$ as the sensed sound source location, with a quality rank of $\hat{Q}$

Fig. 2. *Sound Source Locating Algorithm*

entering the communication subsystem, so as to conserve the scarce wireless bandwidth for effective data. In case of contention resolution, it is desirable to give higher priority to reports with better quality and speed-up their delivery. Inferior reports are assigned lower priority or even dropped in presence of congestion.

To determine the quality of reports during runtime, we use the quality rank defined in acoustic target tracking according to Fig. 2's algorithm. As demonstrated in the experimental result in Section5, quality rank has a strong correlation with the accuracy of the triangulation results.

We implement quality-driven contention resolution and redundancy suppression with the original CSMA MAC protocol in TinyOS. Specifically, we use the quality rank to determine the backoff time for CSMA contention resolution. The better the quality rank, the shorter the backoff. Every time a cluster head/router wants to send out a tracking report packet with a quality rank of Q, its backoff time is computed as,

$$T_{backoff} = Q \cdot interval + random \tag{3}$$

where $interval$ is an implementation-specific constant, Q is the quality rank (with 0 as the highest quality), and $random$ is the random backoff generated by the original CSMA protocol. If before the firing of its own back-off timer, a node overhears a report belonging to the same sound event[1] with a higher quality, it drops its pending report.

The pseudo code of the protocol is described in Fig. 3.

[1] If the Euclidean distance of the two vectors of the acoustic reports $(sound\ source\ location, time)$ is smaller than an error bound, the two reports are regarded as "same sound event"

```
/* Upon generation/reception of an report R(location, time, QualityRank) */
1.  enque(R);
2.  set_backoff_timer(QualityRank*interval+random);
/* Upon overhearing of an report R(location, time) at node i */
3.  if PacketQueue(i) ≠ ∅ {
4.      if find_match(R) ≡ true
5.          drop_inferior_report(QualityRank);
6.  }
/* Upon backoff timer expiration */
7.  R = dequeue();
8.  transmit(R);
```

Fig. 3. *Operation of QDR*

4.2 Sink-tree construction

We propose a multi-parent sink tree routing algorithm. Compared with the original implementation of sink tree routing [5], the main difference is that instead of maintaining a single upstream node (with respect to the data sink), a node keeps a *candidate parent list* of multiple upstream nodes to reach the data sink. The candidate parent list is ordered by certain preference (e.g., the number of hops to the sink). All the parent candidates are maintained as soft states. The data sink periodically sends out flooding packets to construct the sink tree. New parent candidates are inserted into the router's candidate parent list when a flooding packet from the corresponding node is received.

To forward a data packet, a node always tries to forward it to the first parent candidate. If a link failure is detected (e.g. exceeding the retransmission limit), a node drops the corresponding parent candidate and uses the next one in the list.

The main advantage of multi-parent sink tree protocol is that it improves availability by fast recovery. The link failures can be repaired locally using multi-parent information. Compared with the original sink tree routing algorithm, where the links are not repaired until the next round of sink tree building flooding, local recovery can increase the reliability and throughput for packet delivery in the network.

5 Experiments

In this section, we conduct several experiments in our sensor network testbed. Sensors and routers are made up by a number of mica motes. A few pc/104 single board computers are deployed to serve as the cluster heads. Each pc/104 board is connected via serial port to a mica mote for sensing and wireless communication.

5.1 Sound Source Locating within a Cluster

In this section, we study the performance of the acoustic tracking subsystem. Of primary interests are, i) the accuracy of triangulation result and ii) the correlation between triangulation accuracy and quality rank Q proposed in Section 3.

As mentioned earlier, clusters are the basic units for locating and tracking acoustic target. In this set of experiment, sensors are placed uniformly in a $100 \times 100 inch^2$ 2-D area to form a single cluster (see Fig. 4). Three settings are tested, using 8, 10 and 12 sensors per cluster respectively. A pc/104 is placed at the center of the 2-D area to serve as the cluster head. To understand the sensitivity of triangulation result to the sound source location, we also vary the location of sound source as shown in Fig. 4. In each test setting (8, 10 or 12 sensors/cluster), 10 trials are carried out for each of the 18 sound source locations. Therefore, altogether there are $10 \times 18 \times 3 = 640$ trials.

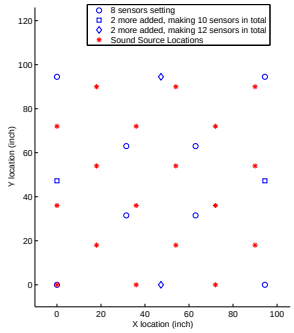


Fig. 4. *Locations of sensors and sound sources in a single cluster*

Accuracy of triangulation The accuracy of triangulation result is defined as the Euclidean distance between the actual location (x_a, y_a) and the computed location (x_m, y_m) as,

$$SensingError = \sqrt{(x_a - x_m)^2 + (y_a - y_m)^2} \tag{4}$$

Fig. 5 gives an example of the tracking results for the 12 sensor case (due to space limit, we only show 6 out of the total 18 locations). Each "$*$" represents the actual location of the sound source, while each "." corresponds to the triangulation result in one trial. As shown in Fig. 5, the majorities of the locating reports fall within the vicinity of the sound source. However, the triangulation accuracy degrades as the sound source moves to the corner of the sensing area. This can mitigated by overlapping multiple sensing clusters.

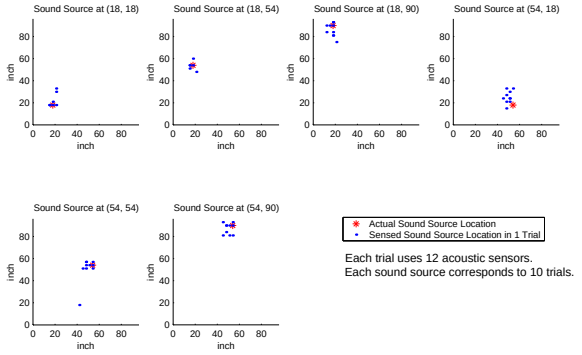
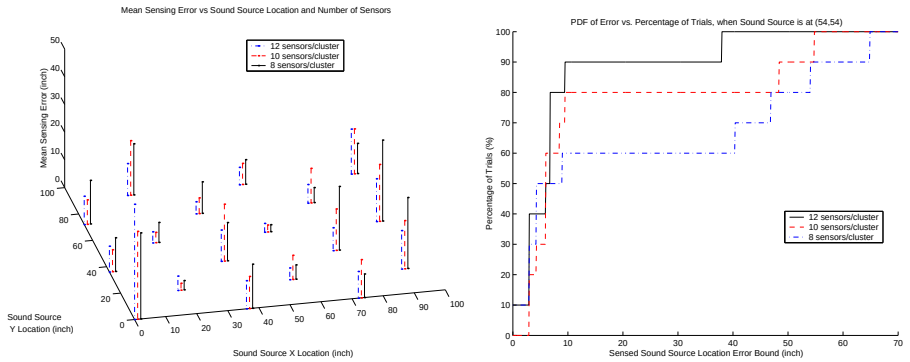


Fig. 5. *An example of triangulation results for different sound source location*



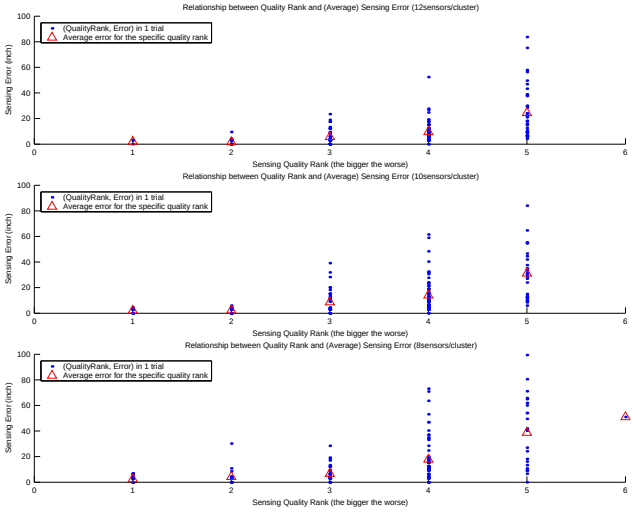
(a) *Average Error vs. sound source locations* (b) *CDF for sensing errors*

Fig. 6. *Distribution of sensing error*

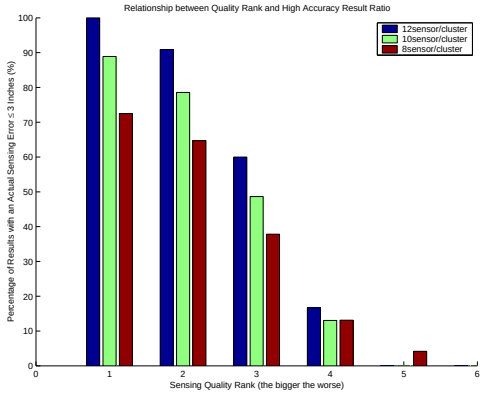
Fig. 6(a) shows the average sensing error with respect to different sound location and the number of sensors used. Roughly speaking, with more sensors, the accuracy of triangulation gets higher. The triangulation result is quite satisfying, considering the fact that the sound source itself (a speaker) is approximately a cubic with dimension 4 inches. There are some cases in Fig. 6(a), where using eight sensor gives the best result. This is because "bad" data generated by multiple sensors can potentially "poison" the result. We will further investigate this issue as one of our future work. A closer view of the stochastic property of the triangulation reports is given in Fig. 6(b) for different number of sensors for a fixed sound source location at (54, 54).

Quality of reports Fig. 7(a) demonstrate the correlation between quality rank and the accuracy of the triangulation result. As expected, the smaller the quality rank (or the higher the quality), the higher the accuracy of the triangulation result. When the quality rank is inferior (with $QualityRank \geq 4$), both the mean and deviation of the sensing errors are very large. On the other hand,

superior quality rank reports are statistically trustworthy. This speaks strongly for using the quality rank defined in Section 3 as an indication of the quality of the triangulation results. In addition, as shown in Fig. 7(b), using more sensors can improve the trustworthiness in superior quality ranks (the percentage of reports at rank 1 within 3-inch error range is close to 100%).



(a) *Quality rank vs. accuracy*



(b) *% of reports within 3-inch error range*

Fig. 7. *Relationship between Quality Rank and accuracy of triangulation results*

5.2 Study of QDR and Multi-parent Sink Tree Routing

In this section, we study the effectiveness of the quality-driven redundancy suppression and contention resolution (QDR). As mentioned earlier, the benefits of QDR scheme for communication subsystem are two-folded, i) it alleviates the channel contention thus leads to higher throughout and ii) it increases the information throughput by giving higher priority for high-quality reports to be transmitted. Therefore, the performance metrics of interests are,

- **The average quality rank of received reports ($\overline{Q}$).** As demonstrated in previous section, the quality rank provide a quantitative measurement of the report. The smaller the quality rank, the better the data quality.
- **Deviation from the minimum rank.** Ideally, only the report with the most superior quality rank should be delivered. However, reports may get lost or multiple reports can be delivered since they are generated at different time. Deviation from the minimum rank defined as the difference from the minimum rank, reflects the effectiveness of communication and suppression.
- **Utility.** In attempt to gauge the rate of effective information throughput (rather than nominal throughput of raw bits), we define a utility function $U(k) = \frac{S_k}{Q_k}$ for the k^{th} packet, where Q_k and S_k are the rank and size of the k^{th} packet.

In this set of experiments, there are 3 ajacent clusters, 7 router motes including the ones attached to the sink and the pc/104 board. A sound source can be heard by all of the clusters. However, depending on the location of the sound source and the triangulation result, different clusters can generate reports of different quality. The reports are routed via a 2-hop communication network to the data sink. Transmission of the reports are subject to the quality-driven backoff timer, which is computed as in Equation 3.

Impact of the backoff timer value: In this set of experiment, we vary the interval from 0ms (no backoff) to 400ms. The percentage of suppressed reports are depicted in Fig. 8. Also shown in the graph are the levels of confidence for five runs of experiments. Consistent with our expectation, as the backoff timer value gets large, more redundant reports get suppressed. However, this comes at the expense of longer interval to deliver the report. Therefore, in the next set of experiment, we choose the interval to be 100ms.

Study of QDR In this set of experiment, we fix the interval to be 100ms and compare the scheme with QDR and one without. From Table 1, we can see that as expected, with the QDR, the average quality of delivered report is better and thus the utility is better. However, the percentage of suppressed reports is not very significant (ideally, it should be around 66.6%). This can be attributed to the fact that the reports are not always generated around the same time. Therefore, a report of inferior quality rank may still get delivered because its backoff period doesn't overlap with the sending of the other reports with superior quality rank.

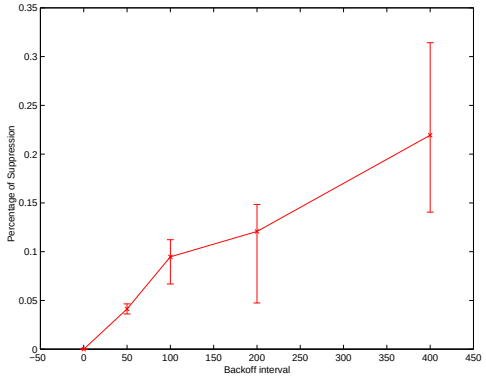


Fig. 8. *Effect of backoff timer on the percentage of suppressed reports*

Table 1. Effect of QDR

	rank	dev. rank	utility	% of dropped reports
without QDR	3.4900	1.2517	0.1160	0
with QDR	3.2920	1.1348	0.1312	9.5%

6 Conclusion and future work

In this paper, we investigate the design and implementation of acoustic tracking using tiny wireless devices. To achieve high reliability, availability in a system of networked sensors with only limited computation and communication capability, we propose decomposition of the different roles and divide the system into two components, i) the acoustic target tracking subsystem and ii) the communication subsystem. Our main contributions can be summarized as follows,

- Designed and implemented an acoustic target tracking system using *tiny* wireless devices.
- Proposed a ranking mechanism to decide the quality of tracking result.
- Proposed the idea of quality-driven redundancy suppression and contention resolution, together with an implementation using quality rank.

Experimental results using our sensor network testbed demonstrate the effectiveness of the proposed design and validate the idea of quality rank.

Through our first-hand experience with the system, we identify several agenda for future work, First, protocol design and experimentation with moving targets needs to be investigated. Of particular interest is the real-time issue to deliver high-quality reports in a timely fashion. Secondly, we are interested in applying energy conservation techniques to the acoustic tracking system. Our hierarchical structure can naturally take advantage of the redundancy in highly dense sensor networks for power saving. Lastly, further study of the quality-driven approach

and its applicability to other application domain for sensor networks should be studied.

References

1. G. C. Carter, Coherence and Time Delay Estimation, IEEE Press, 1993.
2. Jeremy Elson, Lewis Girod, Deborah Estrin,"Fine-Grained Network Time Synchronization using Reference Broadcasts," in Proc. of the Fifth Symposium on Operating Systems Design and Implementation (OSDI2002), Boston, MA. Dec. 2002.
3. D. V. Gadre, "Programming and Customizing the AVR Microcontroller," McGraw-Hill, 2001.
4. L. Girod, V. Bychkovskiy, J. Elson, D. Estrin, "Locating tiny sensors in time and space: A case study," in Proc. of International Conference on Computer Design (ICCD 2002), Freiburg, Germany. Sep., 2002. (Invited Paper)
5. Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David E. Culler, Kristofer S. J. Pister, "System Architecture Directions for Networked Sensors," in Proc. of ASPLOS-IX, Cambridge, Mass. 2000.
6. http://www.jumptec.de
7. http://www.rfm.com/products/data/tr1000.pdf
8. J. Kahn, R. H. Katz, K. Pister, "Emerging Challenges: Mobile Networking for 'Smart Dust'." Journal of Communications and Networks, vol. 2, no. 3. pp. 188-196, Sep. 2000 (Invited Paper).
9. C. W. Reed, et al., "Direct joint source localization and propagation speed estimation," in Proc. of ICASSP'99, Phoenix, AZ, pp. 1169-1172, 1999.
10. B. Parkinson and S. Gilbert, "NAVSTAR: global positioning information system ten years later," in Proceeding of IEEE, pp. 1177-1186, 1983.
11. X. Sheng, Y. Hu, "Energy Based Acoustic Source Localization," in Proc. of the 2nd International Workshop on Information Processing in Sensor Networks, PARC, Palo Alto, CA, Apr. 2003.
12. Jim Waldo, Ken Arnold, et. al., "The Jini Specifications (2nd edition)," Addison-Wesley Pub Co. Dec., 2000.
13. Qixin Wang, "Mote's throughput test report and source code," in http://www.andrew.cmu.edu/ weizhang/wsn/documents.html
14. B. Warneke, B. Atwood, K. S. J. Pister, "Smart Dust Mote Forerunners," in Proc. of the Fourteenth Annual International Conference on Microelectromechanical Systems (MEMES 2001), Interlaken, Switzerland, Jan. 21-25, 2001, pp. 357-360.