# EnviroMic: Towards Cooperative Storage and Retrieval in Audio Sensor Networks *

Liqian Luo[1], Qing Cao[1], Chengdu Huang[1], Tarek Abdelzaher[1], John A. Stankovic[2], Michael Ward[3]
[1]Dept. of Computer Science, Univ.of Illinois at Urbana-Champaign, {lluo2, qcao2, chuang30, zaher}@cs.uiuc.edu
[2]Dept. of Computer Science, Univ. of Virginia, stankovic@cs.virginia.edu
[3]Dept.of Natural Resources & Environmental Sciences, Univ. of Illinois at Urbana-Champaign, mpward@uiuc.edu

## Abstract

*This paper presents* EnviroMic*, a novel distributed acoustic monitoring, storage, and trace retrieval system. Audio represents one of the least exploited modalities in sensor networks to date. The relatively high frequency and large size of audio traces motivate distributed algorithms for coordinating recording tasks, reducing redundancy of data stored by nearby sensors, filtering out silence, and balancing storage utilization in the network. Applications of acoustic monitoring with EnviroMic range from the study of mating rituals and social behavior of animals in the wild to audio surveillance of military targets. EnviroMic is designed for disconnected operation, where the luxury of having a basestation cannot be assumed. We implement the system on a TinyOS-based platform and systematically evaluate its performance through both indoor testbed experiments and a preliminary outdoor deployment. Results demonstrate up to a 4-fold improvement in effective storage capacity of the network compared to uncoordinated recording.*

## 1 Introduction

Most prior sensor network research on environmental monitoring focused on low-bandwidth sensing such as light [1], temperature [19], motion and magnetic fields [12]. Notable exceptions include efforts such as structural monitoring [31], where vibrations were recorded at a frequency of a hundred hertz, and volcano monitoring [29], where sensors deployed near an active volcano sampled seismic and acoustic data at 100Hz. All of the above services, however, assume the availability of a basestation for data uploading during the operation. The *in-network* storage capability of sensor networks remains largely untapped. Another category of environmental monitoring applications focuses on disconnected deployment where data collection occurs only sporadically when researchers drive by the field as in ZebraNet [16] or when the monitored targets approach the basestation as in SATIRE [8]. These applications, however, operate on low-bandwidth sensors that do not produce large data volumes. Comparatively, this paper explores a new direction where the challenges of high-frequency sampling and disconnected deployment coexist in the system. EnviroMic presents the first implementation of an audio sensor network for recording, storing, and retrieving environmental acoustic traces geared for a prolonged interval of disconnected operation.

Acoustic sensors have very little energy requirements and therefore can operate on batteries for long periods. They can be used to detect human speech, geophysical sounds, and distinguish the calls of many birds and animals. Compared to video, acoustic sensing has the advantage of omni-directionality and independence from line-of-sight constraints. Therefore, it presents fewer limitations on sensor placement, and motivates development of acoustic recording services as the first category of multimedia sensing applications, to which EnviroMic belongs.

Data is the most important outcome of an environmental study. EnviroMic is designed to maximize the amount of (acoustic) data collected and stored by the sensor network while disconnected from basestations or other real-time uplinks. Instead, data retrieval is done either by occasionally sending data mules into the field or by physically collecting the sensor nodes after deployment. There are two reasons for considering such disconnected operation. First, many environmental monitoring applications are deployed in harsh unattended environments where access to power is problematic at best, making long-term deployment of high-power data-collection basestations difficult. Second, basestations are centralized points of failure that may be compromised by environmental conditions, animals, thieves, or other (possibly) malicious entities. It is therefore expedient to do away with basestations when real-time response requirements do not mandate their existence.

Current motes are on a trajectory to become viable audio sensing platforms. For example, the original MicaZ mote [3] had only 0.5MB of flash memory. Assuming a sampling rate of 4kHz, that could be supported by the mote, a node would exhaust its local storage within two minutes. With advances in NAND flash, new mote prototypes are now available [22] that interface Mica-class processing and radio hardware to up to 512MB of flash memory; a three orders of magnitude improvement. This trajectory of increasing low-power flash on motes makes it interesting to in-

*The work reported in this paper was funded in part by NSF grants NSF CNS 06-15318, NSF CNS 06-26342, NSF CNS 05-53420 and NSF DNS 05-54759. Any opinions and findings are those of the authors and not necessarily of the funding agencies.

vestigate recording services based on Mica-class hardware. Despite advances in large lower-power flash memory, storage is not an "infinite" resource. For example, an acoustic sensor that has a 1GB flash, if sampling the entire audible spectrum (roughly 20kHz) at the Nyquist frequency (twice the spectrum or 40kHz), will run out of storage in 7 hours. Data storage is therefore a primary concern. Efficient storage management is needed to alleviate this bottleneck in disconnected audio-recording sensor networks.

Networks of acoustic sensors are needed because individual sensors are limited in their effective acoustic range. For example, in a forest, it is difficult to record sounds that occur more than, say, a few hundred feet away. If an area of tens of acres is to be covered, there is no alternative to using multiple microphones. Moreover, for the same reason network capacity is maximized when radio ranges are small [13], deployment of more nodes with smaller acoustic ranges may be preferred to deployment of fewer, more expensive and more sensitive nodes. In the latter case, there is more opportunity for "collisions" (where nearby sounds drown more distant ones), not to mention a lower fidelity in mapping sounds correctly in space.

Networked deployment of large numbers of sensors adds new challenges and opportunities in acoustic recording service design. The main challenges of EnviroMic are enumerated below. First, the omni-directional nature of acoustic sensing introduces data redundancy when multiple nodes collectively sense the same acoustic source. Therefore, EnviroMic should limit redundant recording such that storage is used more efficiently. Second, since current sensor platforms are severely constrained in CPU bandwidth, they are unable to perform other activities (such as radio communication) concurrently with high frequency sampling. Therefore, it is important to coordinate the recording and other tasks such that they do not interfere. Finally, the high data volume generated by an acoustic source, coupled with the potentially uneven spatial distributions of such sources, may cause some nodes to overflow while others still have available storage space. Hence, EnviroMic must balance recorded data across nodes to eliminate storage hot-spots and to make use of storage capacity that is not in direct vicinity of the frequent sound sources.

To address these challenges, we implemented a prototype of EnviroMic using MicaZ motes equipped with MTS300 sensor boards [4]. We chose MicaZ motes because they are compatible with new large NAND-flash extensions (e.g., [22]) and are a good example of low-cost, low-power, and low-range platforms that might host future acoustic services. Due to the lack of availability of large numbers of off-the-shield MicaZ motes with NAND-flash, we used an older readily-available version with a 0.5MB flash. However, the research challenges we address on this prototype are not unique to the hardware platform chosen. Platforms with more storage resources [22][23] still require efficient storage management as argued above. Based on the prototype implementation, we investigated system performance attributes such as data storage redundancy, load balancing and quality of recording. Our evaluation results demonstrate the efficacy of this service in conserving storage resources, preserving the continuity of recordings, and balancing network load.

The rest of the paper is organized as follows. Section 2 presents the system design. Section 3 presents the details for EnviroMic implementation. Section 4 analyzes evaluation results of EnviroMic based on both indoor and outdoor experiments. Section 5 reviews related work. Section 6 concludes the paper.

## 2 System Design

The target application of EnviroMic is long-term acoustic monitoring, which involves high-frequency sampling and high-volume data storage. The primary concern is to fully utilize the effective storage capacity of a sensor network, maximizing the amount of data a scientist can collect about the environment in a single experiment. The network is assumed to be disconnected from the outside world. Hence, there is need for improving storage utilization. With that design goal in mind, we employs mechanisms in EnviroMic to reduce storage redundancy and improve balancing of data storage in the network, when energy permits.

Furthermore, recording in EnviroMic is sound activated. In other words, while sensors are continuously sensing, nothing is recorded unless it exceeds the long-term running average of background noise by a sufficient margin. Because we only record isolated sound clips, a secondary goal is to facilitate indexing of such clips. Specifically, we attempt to store continuous sounds in continuous files as much as possible. We assume that back-end basestations will perform more sophisticated application-specific analysis on data. Such analysis, for example, might include counting bird populations and inferring social communication patterns from isolated vocalizations. Such analysis can also compensate for imperfections in online recording. For example, the basestation might recognize that two files, in fact, refer to the same vocalization. Below, we first briefly highlight the main EnviroMic subsystems.

To reduce storage redundancy and improve load balancing, EnviroMic employs a distributed algorithm that rotates the task of recording local acoustic events among nodes near the source.[1] The recording service, which implements its own specialized file-system, attempts to create a single file for each continuous acoustic event. The file consists of different chunks residing on different sensors that recorded parts of the event. We call this subsystem the *cooperative*

[1]A controlled amount of redundancy can be introduced if needed for robustness, but it is not the focus of this paper.

*recording* subsystem. This subsystem offers two main advantages over uncoordinated local sampling, in which each sensor independently records local acoustic data. First, redundancy is reduced, allowing more data to be collected. Second, recording that is perceived to belong to the same continuous acoustic event is coalesced into the same file. However, the subsystem does not guarantee unique file-to-event mapping; it merely does its best in event-file association to facilitate indexing of sound clips. This provides the basis for more sophisticated algorithms (executed on powerful basestations after collecting all the data) to extract higher-level information.

To further improve storage balancing in the network, EnviroMic transfer data from heavy-loaded nodes to light-loaded nodes when energy permits. Such data transfers are triggered whenever the difference in estimated remaining lifetime (time to overflow) of neighboring nodes, exceeds a certain threshold. We call this service the *distributed storage balancing* subsystem. Finally, a *data retrieval* subsystem provides a simple mechanism for extracting data from the network. These subsystems are described in the following subsections respectively.

## 2.1 Cooperative Recording

Cooperative recording refers to the act of splitting the task of recording an acoustic event among multiple sensors. An inherent assumption is that the average acoustic event of interest will be heard by more than one node. In our protocol, when multiple nodes sense the same acoustic event simultaneously, they form a group. Group members coordinate to elect a leader, who assigns recording tasks to individual nodes that can hear the event. When the acoustic source is a mobile object, group membership may change around the object as it moves. A leader handoff mechanism is employed to preserve the continuity of recording. The rest of this section describe the group management and task assignment mechanisms.

**Group Management**

When an acoustic event occurs, if multiple nodes sense the event within the same locality, they compete to elect a local leader who will ensure that only one copy is recorded. The detailed leader election algorithm is described in our previous work [17]. Briefly, nodes that hear the event start random back-off timers. Upon the expiration of a timer, a node announces leadership unless it has already heard such an announcement from a neighbor. When a leader is elected, it gives a new ID to the current event, which is also the file ID. Observe that multiple leaders may be elected for the same event which will produce redundant recording. Our design choice is not to guarantee complete elimination of redundancy. Instead, our leader election algorithm simply attempts to eliminate redundancy within one-hop communication neighborhoods. This is a compromise between algorithm complexity and performance. With a suitable communication range (e.g., larger than the sensing range for the average acoustic event), redundancy will be eliminated except for very loud acoustic events that are hopefully infrequent.

In our implementation, leader election and assignment of the first recording task take up to one second. Hence, the very beginning of the acoustic event is missed. For relatively long-lasting events such as passage of vehicles and vocalizations of some (singing) species of birds, this startup time is insignificant. For very short acoustic events, we use an optimization to let nodes that hear the event record a configurable small interval (typically one second), called the *prelude*, of each new event locally without coordination. If the event continues past that interval, recording is interrupted to elect a leader and assign recording tasks both for the future and the past. In particular, a node is chosen among those that recorded the prelude to keep the recorded data. All others erase their recording. In this scheme, long-term events are recorded with a brief interruption after the prelude period. Leader election cannot be performed concurrently with recording on the current motes because radio communication can greatly disrupt high-frequency recording due to insufficient CPU capacity of current devices.

Acoustic sources may be stationary or mobile. If a mobile object generates continuous sound along its trajectory, EnviroMic captures this continuity by recording in the same file using a leader handoff mechanism. Specifically, when a leader ceases to sense the acoustic event, it broadcasts a RESIGN message (tagged with the ID of the current event). Upon receiving this message, nodes that can sense the event will compete to be the new leader. The new leader then instructs its members to use the same file ID for their data. File continuity is generally preserved.

It is possible that EnviroMic fails to enforce unique file-to-event association. For an acoustic event with a large spatial signature (such as thunder), more than one leader might be elected as nodes sensing the same event are not within each other's communication range. For overlapping events (e.g., two birds singing next to each other), only one leader may be elected, and thus one file may be recorded. Our goal is merely to reduce redundancy. More sophisticated temporal and spatial correlation algorithms can be performed on these files at basestations to extract more accurate information if needed for the application.

**Task Assignment**

Once a group is formed, the leader is responsible for assigning recording tasks to its group members. While an event lasts, nodes that can hear the event periodically broadcast a SENSING message to notify the neighbors of their awareness of the event. The leader maintains a list of such nodes (called members) and periodically selects one that is most "suitable" for the recording task. It could be the member with the highest time-to-live (see Section 2.2) or the one with the best reception of the acoustic signal.

Once the leader selects a member to assign a recording task to, it sends a TASK_REQUEST message to the member, hereafter called the *recorder*. Note that all the members other than the recorder are listening for control messages. The recorder echoes the request with a TASK_CONFIRM message, and starts recording immediately after the message is successfully sent out. The recording lasts for a predetermined period of time, $T_{rc}$, called *recording task period*. The leader starts a timer waiting for a TASK_CONFIRM from the recorder. Upon successfully receiving a TASK_CONFIRM, the leader schedules the next task assignment to be $T_{rc}$ away. If the leader times out, either the initial TASK_REQUEST or the subsequent TASK_CONFIRM has been lost. The leader immediately selects another member to be the recorder. Note that, this new attempt of selecting a recorder might be caused by a loss of the previous TASK_CONFIRM (instead of the TASK_REQUEST), which implies that a node is already recording. Selecting another recorder in this case may give rise to more than one member recording simultaneously.

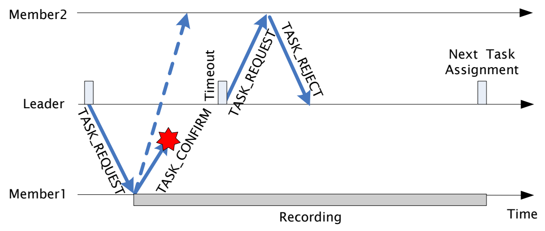


**Figure 1:** Task assignment optimization

We alleviate this problem by an optimization using overhearing, shown in Figure 1. Upon receiving a TASK_REQUEST, the member responds with a TASK_CONFIRM, if it did *not* hear a TASK_CONFIRM earlier. Otherwise, it responds with a TASK_REJECT, because it can infer that some other node is already doing the recording task but the TASK_CONFIRM message was not received by the leader. When a TASK_REJECT or a TASK_CONFIRM message is received by the leader, the leader is assured that the task assignment is done, and hence can schedule the next round of task assignment. This task assignment algorithm minimizes redundant recordings as well as recording misses caused by protocol control packet losses.

Every node that hears the acoustic event maintains a list of members in its neighborhood. This does not, however, incur extra communication cost because all the control packets can be overheard. This soft state maintained in every node is necessary because when leader handoff occurs, the new leader should already have a list of group members to start task assignment right away. The recording continues uninterrupted as long as a new leader takes over and assigns the next recording task before the current task finishes.

## 2.2 Distributed Storage Balancing

To fully utilize available storage capacity, it is essential that EnviroMic eliminates acoustic hot spots in "noisy" regions by pushing data to nodes in "quiet" regions with more unused storage. This load balancing is possible because acoustic events are likely to be sporadic allowing for migration in between occurrences when needed. Scalability and cost concerns lead us to a design where nodes make migration decisions based on local information only. Intuitively, if a node is much more likely to run out of its storage space than its neighbors, it should migrate some of its local data to some neighbors.

Another issue that must be considered in the storage balancing subsystem is energy. Observe that nodes will miss acoustic events after they have saturated their storage or run out of energy, whichever comes first. A node can compute (i) the time when its flash will saturate at the current data acquisition rate if it does not move data out, and (ii) the time when it will run out of energy if it moves data. When an imbalance in storage utilization occurs, a node decides on whether or not to move data based on what maximizes the remaining lifetime (of the two options above). Formally, we use a metric called time-to-live (TTL) to quantify the expected time when a node runs out of its storage space or its energy. We define $TTL_{storage}$, denoting when a node $i$ is expected to run out of storage space, as:

$$TTL_{storage} = \frac{C_i(t)}{R_i(t)} \tag{1}$$

where $C_i(t)$ is the current unused storage of the node, and $R_i(t)$ is the data acquisition rate of node $i$ (when it is awake), measured as the number of bytes recorded over the (waking) interval during which recording took place. It is periodically updated using an exponentially weighted moving average.

We also define the $TTL_{energy}$, denoting when a node $i$ is expected to run out of energy, as:

$$TTL_{energy} = \frac{E_i(t)}{D(R_i(t))} \tag{2}$$

where $D(R_i(t))$ represents the rate of energy drain if the current node moves data out at the acquisition rate $R_i(t)$. It can be easily computed from the idle power consumption, the radio power consumption, and the expected percentage of time radio must be active to transmit at rate $R_i(t)$. The load-balancing sub-system compares $TTL_{storage}$ and $TTL_{energy}$ to determine the appropriate action. If the former is shorter, the current node is allowed to move data out to other nodes. If the latter is shorter, the current node should store data locally. Because load balancing is only triggered when the $TTL_{storage}$ is the bottleneck, we use $TTL$ to refer to the $TTL_{storage}$ unless otherwise stated.

Note that the above computations are completely oblivious to any duty-cycling that the node may be performing. Any duty-cycling will simply extend $TTL_{storage}$ and $TTL_{energy}$ with the same proportion. The bottleneck TTL remains the same.

In the current motes where local storage lasts a few minutes whereas local battery lasts several days, load balancing almost always makes sense. We believe that future mote prototypes will always have spare energy beyond what it takes to simply fill up the local data storage at the expected rate of environmental input. The converse would be questionable from an engineering standpoint. Why bother add so much storage that the battery is not enough to last the time it takes to fill it up?

During their lifetime, nodes monitor their own $TTL$ as well as their neighbors'. Each node updates its state information by local broadcasting. When a node $i$ notices that its $TTL_i$ differs from that of some neighbor $j$ by a certain factor $\beta$:

$$\frac{TTL_j}{TTL_i} > \beta_i \tag{3}$$

and its current $TTL_{energy}$ is larger than its $TTL_{storage}$, it will request to migrate some data to node $j$. Data are transferred from node $i$ to $j$ until condition (3) no longer holds or its $TTL_{energy}$ drops below $TTL_{storage}$.[2] The parameter $\beta_i$ determines how sensitive nodes are to storage imbalance. In practice, we set an upperbound $\beta_{max}$ for $\beta_i$, and let $\beta_i$ varies linearly between 1 and $\beta_{max}$, depending on the current TTL. Observe that nodes are relatively insensitive to imbalanced storage when their TTLs are long, and can become more sensitive as TTLs decrease, which is ideal.

After receiving data transferred from a neighbor, a node might further transfer some of the data to its own neighbors, if necessary. This way, data recorded by nodes in hot-spots can gradually migrate to nodes far away. A relatively balanced storage is hence achieved across the network.

## 2.3 Data Retrieval

In our application scenario, data retrieval occurs very rarely; usually, exactly once when the experiment is over. Hence, reducing retrieval energy does not optimize for the common case. Considering the stringent resource constraints on current motes, we trade some retrieval efficiency for simplicity of design. The inclination was therefore to construct a spanning-tree-based simple broadcast service. User queries specifying the time range and sources that are of interest can be broadcast to the network. The nodes that have files satisfying the query will send these chunks and their file IDs along the spanning tree up to the user.

Further interaction with our system user revealed that commonly the user wished to retrieve the data at the end of the whole experiment when it would be time to physically retrieve the motes from the environment as well. At that point, a single hop version of the aforementioned retrieval scheme was found adequate. When worried about intermediate progress of the experiment, the researcher could enter the sensor network and sample one-hop local measurements. This design provides users a reasonably simple and efficient way to retrieve information.

[2]An alert reader will notice that the scheme makes the implicit assumption that the bottleneck resource tends to be the same on nearby nodes.

## 3 Implementation

We implemented EnviroMic on MicaZ motes running TinyOS. We chose MicaZs since they are good representatives of a large class of off-the-shelf hardware platform. Though MicaZs are severely constrained in storage capacity and may not be the most appropriate for our specific application, the challenges we experienced in building EnviroMic are not specific to MicaZs, but are in fact general to the large class.

The implementation consists of 12 nesC modules, and 10,282 lines of nesC code. The system occupies 61.5KB of code memory and 2.8KB of data memory on MicaZ. Figure 2 gives an overview of the major modules of our implementation, and the interface relationships between them.

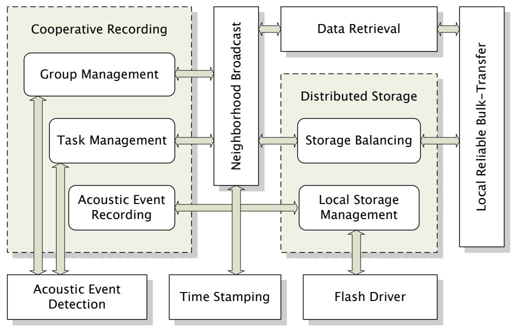


**Figure 2:** Modules of EnviroMic implementation

A few words are in order to describe some modules in Figure 2 that are not discussed in previous sections. Recorded acoustic data are associated with timestamps to ensure they are semantically meaningful. This requires nodes to be (loosely) time-synchronized. Our timestamping module is adapted from FTSP [20]. To make it more power-efficient, we reduce synchronization frequency when events are rare. Besides, clocks at recorders are further synchronized by the receipt of the leader's task assignment messages.

A number of modules in the system require local broadcast, as shown in Figure 2. Messages of some of the modules are delay sensitive (e.g., task management), while messages of some other modules are delay tolerant (e.g., storage balancing). To minimize communication overhead, we implemented a neighborhood broadcast module. All modules that need to do local broadcast register with this module. When a delay sensitive broadcast message is about to be sent out, the neighborhood broadcast module queries all the registered modules to check the possibility of piggybacking some messages from other modules. This mechanism is especially effective when a lot of activities are happening. We also implemented a local reliable bulk-transfer component which is utilized by storage balancing to exchange data between neighbors.

## 4 Evaluation

We evaluate EnviroMic using both an indoor testbed and an outdoor deployment in a forest. The acoustic sampling frequency is set to be 2.730kHz throughout the experiments. The indoor testbed consists of 48 MicaZ motes placed as a 8×6 grid with unit grid length 2ft. We use this testbed together with controlled acoustic events described below to achieve repeatability in our experiments so we could perform valid comparisons and empirically determine the effects of some system parameters. To further understand the performance issues of EnviroMic in realistic environments, we conducted experiments using 36 MicaZ motes in a nearby forest.

### 4.1 Cooperative Recording

Efficiency of the cooperative recording subsystem comprises two related properties. First, we want the acoustic event recording to be complete (i.e., no recording gaps). Second, we want to reduce recording redundancy. Our design and implementation of seamless task assignment (Section 2.1) ensures that recorded data redundancy is almost eliminated. The only (rare) case that could lead to recording redundancy is control packet loss. Hence, in this subsection, we focus on recording misses.

In the implementation of our cooperative recording task assignment mechanism, we introduced the estimated task assignment delay parameter $D_{ta}$ and the task period $T_{rc}$. To empirically determine their values, we used an acoustic mobile target moving through the testbed at a speed of one grid length per second. The event lasts for 9 seconds. The volume was adjusted to set the microphone sensing range of the motes to be about one grid length as well. We ran the experiments 15 times for each combination of the parameters. Figure 3 presents the average and 90% confidence interval of recording miss ratios. The metric *recording miss ratio* is defined as the sum of the lengths of recording gaps divided by the duration of the acoustic event.

From the figure, we can observe that the recording miss ratio first decreases with increasing the expected task assignment delay $D_{ta}$, then levels off after $D_{ta}$ reaches 70ms, and stabilizes at about 8%. Further investigation reveals that this fixed recording miss ratio is due to the initial leader election delay when nodes are forming a group (and no one records). If desired, the initial startup miss can be eliminated by the *prelude* optimization we proposed in Section 2.1.

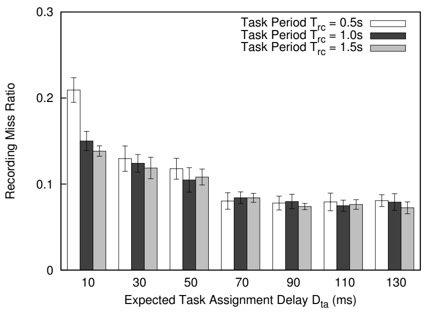
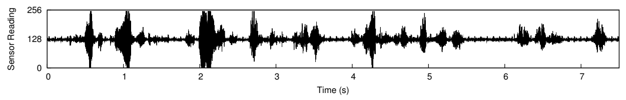
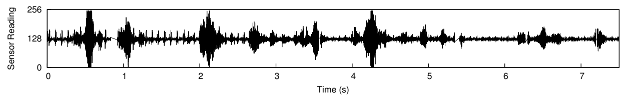


**Figure 3:** Recording miss ratio



(a) Recorded by a single mote



(b) Recorded by EnviroMic

**Figure 4:** Recording voice of a moving human being

Based on the experimental data, 1.0s and 1.5s seem to be good values for the task period. However, we found that the recorded data quality for a task period of 1.5s is noticeably worse than that for a period of 1.0s because the long recording period causes loss of quality when sources are mobile (as the source moves further away from the recorder). Hence, we picked 1.0s as the task period for the rest of our experiments and accordingly 70ms as the expected task assignment delay.

To better appreciate the efficacy of our cooperative recording subsystem, in Figure 4 we present an experiment of recording human voice. In this experiment, a person read out the title of this paper while moving across the 7×4 grid of motes at a constant speed of one grid length per second. An extra mote was held by the person during the experiment to record a reference "ground truth". Figure 4 compares the sensor readings of the mote held by the person with the readings of the nodes that run EnviroMic, stitched together based on their timestamps. The visual similarity of the two figures is obvious. The clips recorded by the single mote and by EnviroMic are available online for qualitative comparison if the reader is interested: `http://www.cs.uiuc.edu/homes/lluo2/enviromic`

### 4.2 Distributed Storage Balancing

In this section, we evaluate the performance of distributed storage balancing.

**Testbed Setup**

In the following experiments, we evaluate EnviroMic using our indoor testbed consisting of 48 MicaZ motes placed as a 8×6 grid (shown in Figure 5). We inject controlled

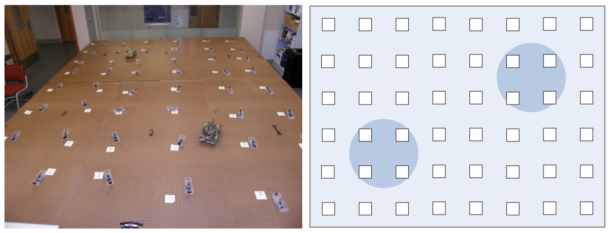


**Figure 5:** Indoor testbed setup

acoustic events into this testbed to achieve repeatability. We use two acoustic sources as event generators that play audio clips. Their locations are shown as shaded circles in Figure 5. All events are generated following a Poisson-distributed event arrival process with an expected inter-arrival time of 20 seconds. The duration of an event follows a uniform distribution between 3 and 7 seconds. Hence, on average, 220 events are generated over a period of 4400 seconds. The average sum of the durations of all events is around 1100 seconds (i.e., 25% of the length of the experiment). To experiment with load balancing, we restrict that only 4 nodes can hear and record each event.

Leaders assign tasks using a $T_{rc}$ of one second, where $T_{rc}$ is the fixed recording task period. We compare different $\beta_{max}$ values in the load balancing subsystem, where we choose values of 2, 3 and 4, respectively. Recall that the actual $\beta$ value varies linearly between 1 and $\beta_{max}$, depending on current TTL. A larger $\beta_{max}$ means that the load balancing subsystem is less sensitive to load imbalance. We also use two baselines. In one baseline, only cooperative recording is used (but without load balancing). In the other, cooperative recording is disabled as well. Each node independently records for $T_{rc}$ upon detecting an acoustic event. These baselines help estimate the total performance improvement of EnviroMic (in terms of reduced acoustic miss ratio).

**Experiment Results**

We evaluate two important metrics of load balancing: recording misses and data redundancy. Recording misses come from two major sources. First, as described in Section 2, it takes a while for nodes to elect a leader and be assigned tasks, upon detecting an acoustic event. We empirically measured this delay to be less than 1 second. In long-term experiments where events are stationary and short, this warm-up time may contribute to a majority of recording misses. Second, when $\beta_{max}$ is high (i.e., when the load balancing sub-system is less sensitive to data distribution imbalances), one node may not balance its load in time to avoid storage overflow, thereby leading to recording misses, especially when its neighbors are also low on remaining storage space. The results of our experiments on recording misses are shown in Figure 6.

From Figure 6, TTL-based load balancing achieves a significantly better performance in terms of miss ratio compared to the two baselines. In these baselines, after the 4 nodes that can detect events fill up their storage spaces, the miss ratio increases considerably. By the end of the experiment, 80% of the data are lost when only local recording is used. With load balancing, the recording miss ratio is much lower. As expected, $\beta_{max} = 2$ achieves the least miss ratio among the different settings because this setting is the most sensitive to load imbalance. By the end of the experiment, less than 20% of the data are lost, which is more than a 4-fold miss ratio improvement. In this case, 4 times more data were recorded with EnviroMic than without. This is of great value, making EnviroMic an attractive research tool for data intensive acoustic studies in biological and environmental science.

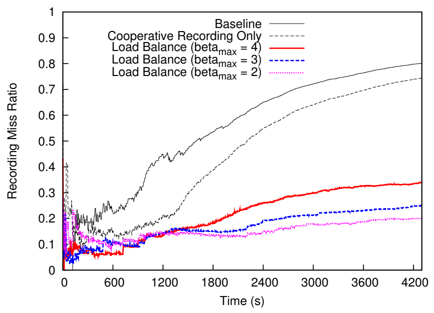


**Figure 6:** Comparison of acoustic recording miss ratio

On the current mote prototype, the improvement in the amount of recorded data comes almost for free in terms of energy. Uploading the entire flash of a MicaZ mote takes less than three minutes. That is an insignificant fraction of lifetime (which is closer to a week). Even if each mote in a "noisy" area was responsible for completely filling up flashes of 10 motes in other areas, the lifetime reduction due to such load balancing should be below one hour. For all practical purposes, it can be ignored. Hence, we did not evaluate the energy cost of load balancing on the current mote prototype.

The second metric we evaluate is the recording redundancy ratio, defined as the ratio between redundant recordings and all recordings. We carried out this experiment us-

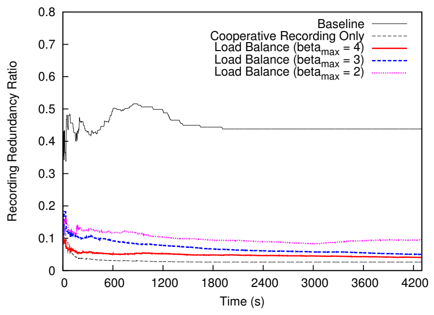
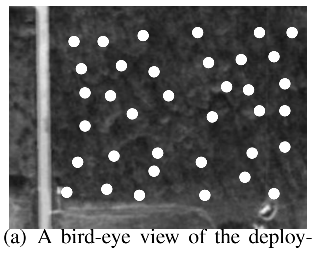


**Figure 7:** Comparison of acoustic recording redundancy ratio



(a) A bird-eye view of the deployment

(b) A mote installed on a tree

**Figure 8:** Deployment in a natural forest

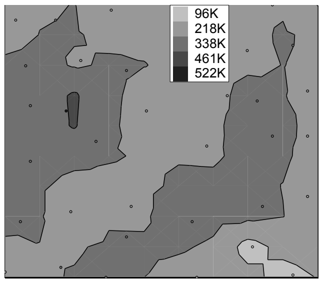


**Figure 10:** Amount of acoustic data, in bytes, generated in different locations

ing the same settings as above, and plot the redundancy ratio in Figure 7. First, observe that the settings where cooperative task assignment is enabled achieve a significantly smaller redundancy compared to the baseline where each node records independently. This observation validates our motivation to design cooperative task assignment as a key strategy to reduce recording redundancy. Second, when $\beta_{max}$ is lower, the observed data redundancy ratio is higher. The reason is that a lower $\beta_{max}$ leads to more data transfers. At last, observe that for the baseline where each node independently records data, the recording redundancy ratio stabilizes around 0.5. This is slightly less than the expected ratio of 0.75 (three out of four traces should be redundant) because individual nodes may not detect the event reliably. Therefore, after one node records for 1 second, it may not detect the event again even if the event persists. This effect goes away in cooperative recording as the odds are high that at least one of the motes surrounding the event will hear it.

## 4.3 Preliminary Outdoor Deployment

To understand the efficacy and limitations of our current design and implementation of EnviroMic in more realistic environments, we deployed a EnviroMic system that consists of 36 MicaZ motes in a forest (Figure 8). On the west side of the forest is a road where vehicles pass by during the day. The experiment was conducted in April 2006. The motes, enclosed in plastic containers, are attached to the trunks of the trees. The deployment area is approximately 105ft×105ft. We were not able to deploy the motes as a grid because the trees in the forest are in irregular positions. We therefore had to reconstruct the map (Figure 8(a)) manually. After the motes were installed, a person holding a mobile device walked around the network to activate the EnviroMic application in each mote. This is to avoid acoustic disturbance caused by installing motes in the containers and attaching them to the trees. We collected data recorded by the motes during a period of 3 hours.

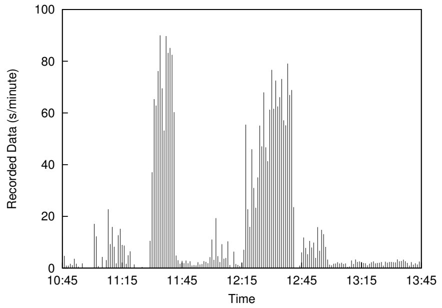


**Figure 9:** Amount of acoustic event data over time

The first set of data of interest is how acoustic events are temporally distributed. Figure 9 plots the amount of acoustic data collected by all the sensors at different times. The numbers on the y-axis represent the total amount of recording (in seconds) done by all nodes within one-minute intervals. They are plotted versus time. There are two spikes in the figure. The first spike (11:30-11:40), as we found out later, was caused by people from another department in our university doing an experiment in the forest. The second spike (12:15-12:45) contains some very long events (up to 73 seconds) that, we conjecture, were caused by the motion of heavy agrarian equipment on a neighboring road.

Next, we look at the geographic distribution of the events. From the data stored in the motes, we reconstructed the information on how much acoustic data was generated from which sensors throughout the experimentation period. Mapping node IDs to geographic locations, we plotted a contour graph of total volume of acoustic events (in seconds) shown in Figure 10. We can see that there are two high data volume regions. The one on the left side is caused by vehicles passing on the road to the west of the forest. The other region that is rich in acoustic events in the figure roughly matches a trail in the forest.

## 4.4 Long-term Deployment Plan

Given the encouraging results obtained from preliminary outdoor testing, we plan to do a long-term deployment of EnviroMic. The objective is a study in avian ecology. In particular, we explore when and where birds vocalize. There are two questions related to bird vocalizations that remain largely unaddressed; why diurnal birds sing at night and what the function of dawn chorus is. Although it has been known for years that many diurnal species vocalize at night, little data, and few hypotheses have been put forth to explain this phenomenon. The reason for the paucity in data is the cost and logistics of recording vocalizations at night. EnviroMic provides an ideal way to record the species that are vocalizing, determine how often different species vocalize, and identify whether or not there is a temporal aspect to nocturnal singing. For example, one hypothesis for why diurnal birds sing at night is that males sing to attract females as they migrate over (most migratory birds migrate at night). The aforementioned study can verify this hypothesis.

Another prospective study enabled by EnviroMic is the study of dawn chorus. Dawn chorus refers to the fact that many species sing at the highest rate at dawn. There are several hypotheses for why this may occur, some relating to the quality of a male and some to the quality of the habitat a male is occupying. The EnviroMic can help determine the environmental influences related to this question.

In general, EnviroMic can be used to investigate social and mating behavior of species such as small mammals, insects, and amphibians that are located in areas with limited access. Several such studies are currently being planned.

## 5 Related Work

Sensor network applications have used acoustic sensors for different purposes, including localization [26], surveillance [12], communication [28], and geophysical monitoring [29]. None of these applications let users retrieve raw acoustic sensor samplings. Comparatively, in EnviroMic, we store raw sampling results in a cooperative manner into local storage, and retrieve them later upon user request. The way we handle data remotely echoes data-mule [25], which also uses a store-and-fetch model.

For acoustic systems, challenges arise to handle the high data volume generated by high frequency sampling of acoustic sensors. To tackle the problem, EnviroMic mainly focus on reducing data redundancy. Obviously, other techniques including in-network filters [11] and data compression algorithms [24] can be easily integrated into EnviroMic to further reduce the data volume to be stored in network.

Also, EnviroMic has a great potential to be applied in applications which previously did not use acoustic sensors. For example, in animal monitoring [19][16] EnviroMic may characterize the behavior of animals from perspectives different from previous approaches using temperature or GPS sensors: acoustic data are much richer in nature and provide direct reflections of animal behavior. Furthermore, data retrieved by EnviroMic can be correlated with data from other sensors to reveal hidden behavior patterns. The authors of [14] implemented an acoustic sensor network application that monitors cane toads. However, they assume the existence of more storage-rich devices for real-time data uploading.

Storage services have been proposed for individual nodes ([5][9][32][21]) as well as networks of distributed nodes ([6][7][27]). However, none of these storage services is appropriate for EnviroMic because of its unique challenge of achieving spontaneous and cooperative storage. To achieve this purpose, we use a leader election algorithm similar to the model presented by EnviroTrack [17], where leaders are elected dynamically to coordinate and assign recording tasks to non-leader nodes.

EnviroMis is also different from our previous work EnviroStore [18], a cooperative distributed storage service. EnviroStore focuses on how to take the best advantage of uploading opportunities when data mules [25] become close.

One key challenge of our distributed storage service is load balancing. Load balancing has been used for other purposes in sensor networks, including maximizing system lifetime by balancing energy consumption of different nodes [15], and improving fairness by balancing MAC layer accesses [30]. Load balancing in EnviroMic is similar to the former, where the available storage space is similar to the remaining battery for each node. However, previous energy load-balancing algorithms can not be directly used for EnviroMic because when applied to storage, we have the additional control knob of exchanging data between nodes, which is impossible in energy-centered load balancing since nodes can not charge each other using their own batteries.

More broadly, load-balancing comprises many algorithms that are studied in different application contexts. Representative applications include load-balancing in web servers [2] and P2P networks [10]. These applications commonly involve many nodes, each with a finite resource capacity. When more resource (bandwidth, computing power, etc.) than desired is consumed, this node tries to reduce its resource consumption by transferring some load to its peers. While this general description also holds for EnviroMic, there are considerable differences. First, EnviroMic has a much higher cost associated with load transfer compared to other load balancing applications, where the load transfer cost is usually sufficiently small [2][10]. In EnviroMic, the energy consumption to transfer acoustic recordings between nodes is usually higher than the energy consumption to write such recordings into flash. Therefore, EnviroMic must explicitly take into account this aspect and make load-balancing decisions based not only on resource consumption, but on energy consumption as well. Second, because of the limited resource limitations of an individual node, no single node is able to coordinate all the other nodes. Therefore, load balancing in EnviroMic must be distributed, and be scalable in the face of the size of the network.

## 6 Conclusion

In this paper, we presented EnviroMic, a distributed acoustic monitoring, storage, and trace retrieval system. The long-term disconnected service model for our target applications calls for a design which stores recorded acoustic data in the network. EnviroMic employs a cooperative recording scheme and a distributed balanced storage mechanism to address unique challenges arising from high-frequency acoustic sampling and high-volume sensory data storage. Data chunks recorded are tagged with timestamps, node IDs, and event (file) IDs to facilitate data retrieval. EnviroMic is implemented on MicaZ motes running TinyOS. Evaluation results drawn from both indoor and outdoor deployments demonstrate the efficacy of our design. Significant system (storage) lifetime improvement is observed compared to baseline algorithms at a modest overhead.

We are currently working on a large-scale long-term deployment of EnviroMic for bird vocalization monitoring and recording. We plan to investigate more intelligent storage balancing algorithms, such as data compression and global (as opposed to local greedy) load-balancing. In a long-term deployment, reliability is apparently a concern. Defunct or lost motes can cause data loss. In this case, a controlled data redundancy may become desirable.

## References

[1] M. A. Batalin, M. Rahimi, Y. Yu, D. Liu, A. Kansal, G. S. Sukhatme, W. J. Kaiser, M. Hansen, G. J. Pottie, M. Srivastava, and D. Estrin. Call and response: experiments in sampling the environment. In *SenSys*, 2004.

[2] V. Cardellini, M. Colajanni, and P. S. Yu. Dynamic load balancing on web-server systems. In *IEEE Internet Computing*, 1999.

[3] Crossbow Technology Inc. micaz motes, 2006. `http://www.xbow.com`.

[4] Crossbow Technology Inc. MTS300 Multi Sensor Board, 2006. `http://www.xbow.com`.

[5] H. Dai, M. Neufeld, and R. Han. Elf: an efficient log-structured flash file system for micro sensor nodes. In *SenSys*, 2004.

[6] P. Desnoyers, D. Ganesan, and P. Shenoy. Tsar: a two tier sensor storage architecture using interval skip graphs. In *SenSys*, 2005.

[7] D. Ganesan, B. Greenstein, D. Perelyubskiy, D. Estrin, and J. Heidemann. An evaluation of multi-resolution storage for sensor networks. In *SenSys*, 2003.

[8] R. K. Ganti, P. Jayachandran, T. F. Abdelzaher, and J. A. Stankovic. Satire: a software architecture for smart attire. In *MobiSys*, 2006.

[9] D. Gay. Matchbox: A simple filing system for motes, 2003. `http://www.tinyos.net/tinyos-1.x/doc/matchbox.pdf`.

[10] B. Godfrey, K. Lakshminarayanan, S. Surana, R. Karp, and I. Stoica. Load balancing in dynamic structured p2p systems. In *IEEE Infocom*, 2004.

[11] B. Greenstein, C. Mar, A. Pesterev, S. Farshchi, E. Kohler, J. Judy, and D. Estrin. Capturing high-frequency phenomena using a bandwidth-limited sensor network. In *SenSys*, 2006.

[12] L. Gu, D. Jia, P. Vicaire, T. Yan, L. Luo, A. Tirumala, Q. Cao, T. He, J. A. Stankovic, T. Abdelzaher, and B. H. Krogh. Lightweight detection and classification for wireless sensor networks in realistic environments. In *SenSys*, 2005.

[13] P. Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, March 2000.

[14] W. Hu, V. N. Tran, N. Bulusu, C. T. Chou, S. Jha, and A. Taylor. The design and evaluation of a hybrid sensor network for cane-toad monitoring. In *IPSN*, 2005.

[15] Q. Li, J. Aslam, and D. Rus. Online power-aware routing in wireless ad-hoc networks. In *Mobicom*, 2001.

[16] T. Liu, C. M. Sadler, P. Zhang, and M. Martonosi. Implementing software on resource-constrained mobile sensors: experiences with impala and zebranet. In *MobiSys*, 2004.

[17] L. Luo, T. Abdelzaher, T. He, and J. Stankovic. Envirosuite: An environmentally immersive programming framework for sensor networks. In *ACM Transactions on Embedded Computing Systems (TECS)*, 2006.

[18] L. Luo, C. Huang, T. Abdelzaher, J. A. Stankovic, and X. Liu. Envirostore: A cooperative storage system for disconnected operation in sensor networks. In *INFOCOM*, 2007.

[19] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson. Wireless sensor networks for habitat monitoring. In *WSNA*, 2002.

[20] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi. The flooding time synchronization protocol. In *SenSys*, 2004.

[21] G. Mathur, P. Desnoyers, D. Ganesan, and P. Shenoy. Capsule: An energy-optimized object storage system for memory-constrained sensor devices. In *SenSys*, 2006.

[22] G. Mathur, P. Desnoyers, D. Ganesan, and P. Shenoy. Ultra-low power data storage for sensor networks. In *IPSN/SPOTS*, 2006.

[23] A. Mitra, A. Banerjee, W. Najjar, D. Zeinalipour-Yazti, V. Kalogeraki, and D. Gunopulos. High-performance, low-power sensor platforms featuring gigabyte scale storage. In *SenMetrics*, 2005.

[24] C. M. Sadler and M. Martonosi. Data compression algorithms for energy-constrained devices in delay tolerant networks. In *SenSys*, 2006.

[25] R. C. Shah, S. Roy, S. Jain, and W. Brunette. Data mules: Modeling a three-tier architecture for sparse sensor networks. In *SNPA*, 2003.

[26] G. Simon, M. Maroti, A. Ledeczi, G. Balogh, B. Kusy, A. Nadas, G. Pap, J. Sallai, and K. Frampton. Sensor network-based countersniper system. In *SenSys*, 2004.

[27] S. Tilak and N. B.Abu-Ghazaleh. Collaborative storage management in sensor networks. In *International Journal on Ad Hoc and Ubiquitous Computing, Vol 1, Nos. 1/2*, 2005.

[28] I. Vasilescu, K. Kotay, D. Rus, P. Corke, and M. Dunbabin. Data collection, storage, and retrieval with an underwater sensor network. In *SenSys*, 2005.

[29] G. Werner-Allen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh. Fidelity and yield in a volcano monitoring sensor network. In *OSDI*, 2006.

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

[31] N. Xu, S. Rangwala, K. K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin. A wireless sensor network for structural monitoring. In *SenSys*, 2004.

[32] D. Zeinalipour-Yazti, S. Lin, V. Kalogeraki, D. Gunopulos, and W. A. Najjar. Microhash: An efficient index structure for flash-based sensor devices. In *FAST*, 2005.