EnviroMic: Towards Cooperative Storage and Retrieval in Audio Sensor Networks

Liqian Luo*, Qing Cao*, Chengdu Huang*, Tarek Abdelzaher*, John A. Stankovic†, Michael Ward‡

*Dept. of Computer Science, Univ.of Illinois at Urbana-Champaign, {lluo2, qcao2, chuang30, zaher}@cs.uiuc.edu

†Dept. of Computer Science, Univ. of Virginia, stankovic@cs.virginia.edu

‡Dept.of Natural Resources & Environmental Sciences, Univ. of Illinois at Urbana-Champaign, mpward@uiuc.edu

Abstract

This paper presents the design, implementation, and evaluation of *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.

Index Terms

Sensor networks, applications, acoustics, distributed storage, group management

I. INTRODUCTION

Most prior sensor network research on environmental monitoring has focused on low-bandwidth sensors such as light [1], temperature [20], motion and magnetic fields [12]. Notable exceptions include efforts such as structural monitoring [32], where vibrations in a building were recorded at a frequency of a hundred hertz, and volcano monitoring sensor network [30], where sensors deployed near an active volcano sampled seismic and acoustic data at 100Hz. In such systems, the sensors (including special-purpose hardware) were used to sample and report non-scalar data to a basestation, leading to challenges in communicating and correlating bursty measurements. 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 [17] 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, which means that they can operate on batteries for prolonged periods of time. They can be used to detect human speech, geophysical sounds, and distinguish the calls of many birds and animals.

The authors are an interdisciplinary team interested in studying the spatial density, frequency, and patterns of vocalizations of different species to learn about their social behavior, mating, and predatory habits. 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 during a prolonged interval of disconnected operation. Disconnected operation refers to the fact that no basestation or other real-time uplink is assumed that can drain data continuously from the sensor network. 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 this mode of 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 induced 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.

Despite advances in large lower-power flash memory, storage is not an "infinite" resource in disconnected acoustic sensor networks. For example, an acoustic sensor that has a 1GB flash and is designed to sample 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. 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 [23] that interface Mica-class processing and radio hardware to up to 512MB of flash memory; a three orders of magnitude improvement. This is enough for days of *continuous* recording (at 4kHz). If recording occurs only when sound is present, the recording lifetime can be further significantly extended. This trajectory of increasing low-power flash on motes makes it interesting to investigate recording services based on Mica-class hardware.

Networks of acoustic sensors are needed because of sensitivity limitations of individual sensors that limit 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 several 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. The spatial fidelity problem is because sounds can originate anywhere in the sensor range (and hence, a larger range creates more uncertainty). Angle-of-arrival techniques [25] cannot be used well in forests due to multi-reflections.

Networked deployment of large numbers of low-cost, low-range sensors adds new challenges and opportunities in acoustic recording service design. The main challenges in the design and implementation 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, it is imperative to ensure that EnviroMic limits 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., [23]) and are a good example of a low-cost, low-power, and low-range platform that might host future acoustic services. Due to the lack of availability of large numbers of MicaZ motes with NAND-flash, we used an older readily-available version with only a 0.5MB flash. Note that the research challenges we address on this prototype, however, are not unique to the hardware platform chosen. Platforms with more storage resources [23][24] 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 II presents the system design. Section III presents the details for EnviroMic implementation. Section IV analyzes evaluation results of EnviroMic based on both indoor and outdoor experiments. Section V reviews related work. Section VI concludes the paper.

II. 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 of the system is to maximize the effective storage capacity of a sensor network, which is maximizing the amount of data a scientist can collect about the environment in a single experiment. It is assumed that the network generally remains disconnected from the outside world during the experiment. Hence, there is need for improving storage capacity. With that design goal in mind, EnviroMic employs mechanisms 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, EnviroMic employs a distributed algorithm that identifies local acoustic events and rotates the task of recording such events among nodes near the perceived source (for load balancing purposes).[1] The recording service, which implements its own specialized file-system, attempts to create a single file for each continuous acoustic event. The file is distributed and consists of different chunks residing on different sensors that recorded parts of the event. We call this subsystem the *cooperative recording* subsystem. The cooperative recording 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, it is not an objective of the subsystem to guarantee unique file-to-event mapping. It merely does its best in event-file association to reduce redundancy and 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 improve storage balancing in the network, EnviroMic further allows nodes to transfer data from heavy-loaded areas to light-loaded areas 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 the subsystem implementing this service, a *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.

A. 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. While one may argue that it is possible to deploy microphones far enough apart that they do not hear the same events, it is impossible to choose the right sensitivity due to the huge variance between signal strength of different acoustic events (e.g., bird singing vs. thunder). Deploying high-sensitivity microphones sparsely leads to loss of spatial granularity and more opportunity for "collision" (multiple acoustic events are collapsed into one), while deploying microphones densely leads to duplicate observations. To err on the safe side, it is better to allow some redundancy and eliminate duplicate observations in software. Finally, note that acoustic ranges of microphones are not polygons that can be tiled without overlap. Hence, even if redundant coverage was not a goal, simply ensuring full coverage of an entire area entails that many points will lie in overlap regions within reach of multiple sensors.

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.

1) 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 used for

---

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

this purpose is described in our previous work [18]. 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. The leader election process is local and does not involve multihop communication. Hence, leaders are associated with local (i.e., single-hop) neighborhoods in which the event was heard. 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 choice of 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 a total of 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, this approach might lead to non-negligible recording misses. Therefore, 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. The length of the prelude can be chosen such that short-term events are fully recorded with high probability. 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 (an issue we will revisit in Section III-B.1). On faster devices, it is possible to overlap prelude recording with leader election, hence resulting in uninterrupted acoustic records for both short and long events.

Acoustic sources may be stationary or mobile. If a mobile object generates continuous sound along its trajectory, this continuity is captured by recording in the same file. EnviroMic has a leader handoff mechanism to handle this issue. Specifically, when a leader ceases to sense the acoustic event, it broadcasts a RESIGN message. Upon receiving this message, nodes that can sense the event will compete to be the new leader. Note that the ID of the current event is attached in the RESIGN message. Therefore, the new leader can instruct its members to use the same file ID for their data. File continuity is generally preserved.

It is possible for more than one leader to be elected for a single event, especially when nodes sensing the same event are not within each other's communication range. An acoustic event with a large spatial signature (such as thunder) may be associated with multiple leaders and thus multiple files. 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. Finally, a temporally separated event (e.g., intermittent vocalizations of the same bird) may give rise to multiple files. Our goal is merely to reduce redundancy such that storage capacity is maximized. 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.

2) 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. Hence, the leader knows which neighbors can be assigned a recording task. The leader periodically selects a node from its member list (i.e., a node that can sense the event) that is most "suitable" for the recording task. It could be the member that has the highest time-to-live (see Section II-B) or the one that has 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. Recorded data are labeled with the current timestamp, the node ID, and event (i.e., file) ID specified by the leader, and stored in local flash. 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.

We alleviate this problem by an optimization using over-hearing, 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 message, 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
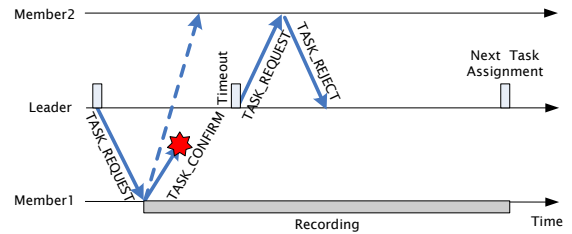

Fig. 1. Task assignment optimization

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. Observe that it is still possible that more than one recorder exist simultaneously because not all the members can always hear each other due to lack of reliable communication. In the scenario depicted in Figure 1, if the TASK_CONFIRM sent by $Member_1$ was not heard by $Member_2$, both nodes would end up recording simultaneously.

Every node that hears the acoustic event (as opposed to the leader only) maintains a list of nodes that also hear the event (i.e., send SENSING messages) 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.

B. 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. Given the current energy level of a node as well as the recorded average input rate due to local acoustic sources, the node can compute (i) the time when its flash will saturate at this 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:

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

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:

$$R_i(t) = R_i(t-1) \cdot (1 - \alpha) + r \cdot \alpha$$

where r is the amount of newly recorded data per unit time during the last period).

The initial data acquisition rate R_0 can be set to zero, or inferred from an estimated average acoustic data generation rate $Exp(R_{event})$ (i.e., the average amount of data generated by acoustic events across the network per unit time): $R_0 = \frac{Exp(R_{event})}{N}$, where N is the number of nodes in the network. This R_0 is basically the average data acquisition rate if acoustic events are uniformly distributed among the nodes. An inaccurate R_0 has little impact on the system behavior in the long run because the data rate is dynamically adjusted.

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

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

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}, and uses the bottleneck (the one that is shorter) 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, in the rest of this section, 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. This is

acceptable because when the node is asleep, neither flash not energy is consumed. Furthermore, $R_i(t)$ refers to the rate of data input from the environment when the node is *awake*. Consequently, 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'. To enable nodes to maintain relatively up-to-date views of their neighborhoods, nodes need to update their state information by local broadcasting. This requirement does not incur much extra overhead because the group management component in Section II-A.1 already maintains soft states for neighbors. When a node i notices that its TTL_i differs from that of some neighbor j by a certain factor β:

$$\frac{TTL_j}{TTL_i} > \beta_i \qquad (1)$$

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 (1) no longer holds or its TTL_{energy} drops below $TTL_{storage}$.[2] The parameter β_i determines how sensitive nodes are to storage imbalance. In practice, we set an upperbound β_{max} for β_i, and let β_i varies linearly between 1 and β_{max}, depending on the current TTL. The larger the β_i, the less sensitive the nodes are. 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.

C. Data Retrieval

In designing a data retrieval algorithm to extract acoustic data from the network, our first intuition was to construct doubly-linked lists for each file to allow efficient retrieval once one chunk of the file was found. Together with some distributed indexing algorithm, the scheme would have optimized retrieval efficiency.

However, as mentioned above, EnviroMic targets long-term, unattended, and disconnected acoustic monitoring and storage applications. Data retrieval occurs very rarely; usually, exactly once when the experiment is over. Hence, reducing retrieval energy does not optimize for the common case. Our next inclination was therefore to trade some retrieval efficiency for simplicity of design. Considering the stringent resource constraints on current motes, a simple design is very desirable even at the expense of some loss of performance.

The inclination was therefore to construct a spanning-tree-based simple broadcast service. Similar to directed diffusion [15], the spanning tree is rooted at the user (the sink). Recall that recorded data are labeled with timestamps, node IDs (which may be mapped to locations if localization services are available), and event IDs. User queries specifying the time range and

---

[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.

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. If gaps are observed in retrieved files, their IDs are flooded until all parts are retrieved successfully.

Further interaction with our system user revealed that the most common query would be one that simply retrieved all files, suggesting further reductions in complexity. Moreover, 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, the need for a multihop broadcast seemed questionable. A single hop version of the aforementioned retrieval scheme was found adequate, since data would usually be retrieved when the motes were physically back in the researcher's lab. When worried about intermediate progress of the experiment, the researcher could enter the sensor network and sample one-hop local measurements.

This design proved to be robust, easy to implement, and at the same time, provided users a reasonably simple and efficient way to retrieve information. It raises the more general question of the future of data collection when low-power storage capacity increases at a higher rate than low-power communication bandwidth. As the gap widens, solutions that minimize communication (e.g., ones based on data mules) may be increasingly preferred. In our case, the user acts as the data mule when they physically collect the motes at the end of the experiment.

III. IMPLEMENTATION

A. Implementation Overview

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 platforms with various sensing and storage capabilities. 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.
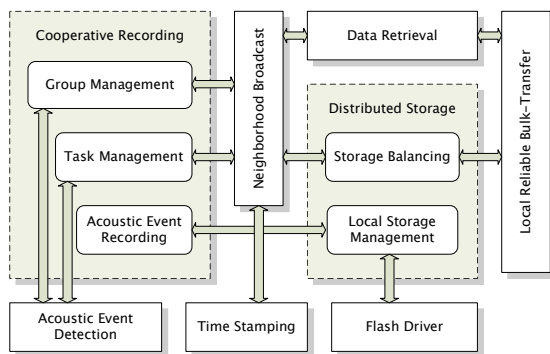

Fig. 2. Modules of EnviroMic implementation

A few words are in order to describe some modules in Figure 2 that are not discussed in previous sections. Recall that recorded acoustic data are associated with timestamps to ensure they are semantically meaningful. This requires nodes to be (loosely) time-synchronized. Our time-stamping module is adapted from FTSP [21]. 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.

B. Implementation Issues

In this subsection we present a few technical issues we encountered in our implementation, and our solutions to these problems.

1) Acoustic Sampling: To record acoustic data with reasonable quality, nodes need to do acoustic sampling at a high frequency. However, with very limited CPU capacity, nodes can not simultaneously perform a high-sampling-rate recording job and a communication task. Hence, we do not allow a node to communicate when recording. When it comes to our implementation, it is also important to make sure that recorders do not experience disturbance from receiving packets for it can significantly slow down the acoustic sampling as well.

Figure 3 illustrates this effect. In this experiment, we set the sampling intervals to 10 jiffies (1 jiffy is 1/32768 second), and measured the real intervals over time by time-stamping each sample, using MicaZ motes. Figure 3 shows the observed intervals for different scenarios.

In Figure 3(a), the node is exclusively sampling data. Therefore, the observed sampling intervals are fixed at 10 jiffies. In Figure 3(b), the node sends out a packet as soon as it starts sampling. Observe that the sampling interval now jumps between 9 and 16 jiffies. Similarly, when a node receives a packet in Figure 3(c), we observe jitter in the sampling interval. Note that this jitter is observed even though the application layer does not process the incoming packets. The reason is that the underlying radio layer still consumes CPU cycles to process packets whenever radio activities are detected.

When a node becomes a recorder, it therefore turns off the radio transceiver completely during the recording task. It is possible that a recorder misses some protocol control messages (e.g., neighbor state updates). Therefore, when the

(a) No communication

(b) Sending a packet

(c) Receiving a packet

Fig. 3. Measured sampling interval between consecutive samples

node turns on its radio again, some of its state information is stale. We choose not to synchronize state at this point to simplify design and conserve energy. Observe that completely up-to-date state information is not required in our protocol.

2) Seamless Task Assignment: As discussed in Section II-A.2, a leader assigns recording tasks to its group members to achieve continuous recording of events. Since each successful recording task lasts for a fixed interval, T_{rc}, conceptually, leaders
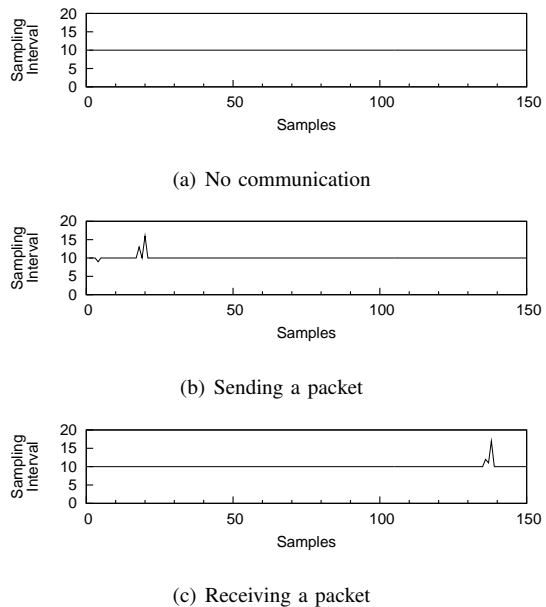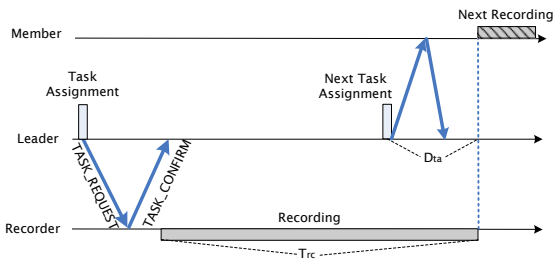
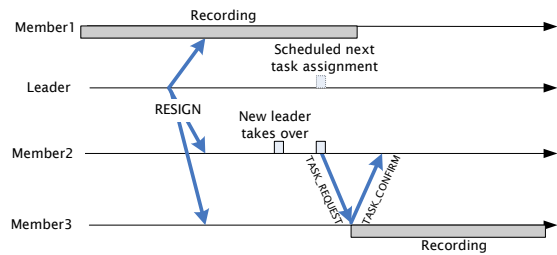Fig. 4. Expected task assignment delay D_{ta} is necessary for seamless task assignment

Fig. 5. Leader handoff process is unlikely to cause recording misses

should schedule a new task (i.e., a new recording node) every T_{rc}. In the implementation, we observe that task assignment itself actually takes time. In order to achieve a seamless transition from one recording node to another (i.e., ensure that there is no gap between consecutive recording intervals), the leader must initiate a new round of task assignment before T_{rc} expires. In practice, we let the leader initiate a new task assignment at $T_{rc} - D_{ta}$ after the previous assignment, where D_{ta} is the expected task assignment delay, shown in Figure 4. Although D_{ta} can be affected by current traffic conditions, we find that using an empirically determined fixed value suffices in our case. The reason is that traffic that can affect D_{ta} has very little variation: the group management and task management packets are pretty regular, and storage balancing traffic is rare.

Obviously, an underestimated task assignment delay can cause recording misses. However, a significantly overestimated task assignment delay is also undesirable. This is because starting a task assignment too early (due to an overestimated task assignment delay) can potentially make the leader's selection of the recording node inferior, since it will be based on older information. By the time that node's turn comes to record, the acoustic source may have moved away and can be better heard by a different node. Hence, we want a relatively accurate estimated task assignment delay. We shall further evaluate the impact of choosing a different D_{ta} in the evaluation section.

Note that leader handoff is very unlikely to cause recording misses because of the way task management works. As shown in Figure 5, when the current leader resigns (because it can not hear the event anymore), a recording task is still ongoing. In the RESIGN message sent out by the resigning leader, the scheduled next task assignment time is attached, so that the new leader can start its first task assignment at the right time. As long as the leader handoff process, which is very quick, finishes before the previously scheduled next task assignment time, no recording gap should occur.

3) Local Data Organization: The local storage space of a node (flash memory) is broken into fixed-length blocks (of 256 bytes in our current implementation), organized as a circular buffer. Data acquired by a node from its sensor or received from neighbors are stored as fixed-sized storage blocks. The stored data chunks form a circular queue. Incoming data chunks are enqueued at the tail of the queue. Data chunks sent to neighbors (to balance storage) are taken from the head of the queue. The benefit of this implementation is that all the blocks receive almost the same number of write operations (different by at most 1). This is desirable because flash memory has write limits. We periodically save the head and tail pointers of the queue to the in-chip EEPROM of MicaZ motes, which has a much larger write limit, so that even if a node fails we can still correctly retrieve its locally stored data after the node is collected.

Each data chunk is associated with certain metadata, including start and end timestamps, a location-stamp (or the ID of the recording node which can be translated to location in a data post-processing stage), and an event (i.e., file) ID. Such information

is necessary for serving data retrieval requests in the future. Note that data chunks stored at a node are not necessarily recorded by the node itself. They could be transferred from a remote node that was short of storage. With the time- and source-stamps attached to each data chunk, a basestation can easily figure out activities in the monitored area across time and space. The event ID further identifies a series of chunks that correspond to a single continuous event as far as EnviroMic could tell.

IV. 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.

A. 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 II-A.2 and III-B.2) ensures that recorded data redundancy is almost eliminated. The only (rare) case that could lead to recording redundancy is control packet loss. As mentioned in Section II-A.2, losses of TASK_CONFIRM may cause multiple nodes to record simultaneously. Hence, in this subsection, we focus on recording misses.

Recall that in the implementation of our cooperative recording task assignment mechanism we introduce the estimated task assignment delay parameter, D_{ta}. Estimated task assignment delay represents how far in advance of the termination of the current recording task a leader should start assigning a new one. As mentioned in Section III-B.2, a D_{ta} that is too small can cause recording misses, while one that is too large can potentially cause leaders to select inappropriate recorders (those that no longer hear the acoustic source).

Similar to the estimated task assignment delay, there is a trade-off in deciding the value of the task period T_{rc}. A shorter task period means more task assignments for the same amount of acoustic events, and hence a higher control overhead. On the other hand, a large task period has a problem with mobile acoustic events. When an acoustic event is present, the node performing the recording task should be able to hear the event clearly. If the task period is too long, the acoustic object might leave the sensing range of the node currently recording before the expiration of its recording interval.

To study these trade-offs and empirically determine the values of D_{ta} and T_{rc}, we used an acoustic mobile target moving through the testbed at a speed of one grid length per second. The event lasts for a total of 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 6 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
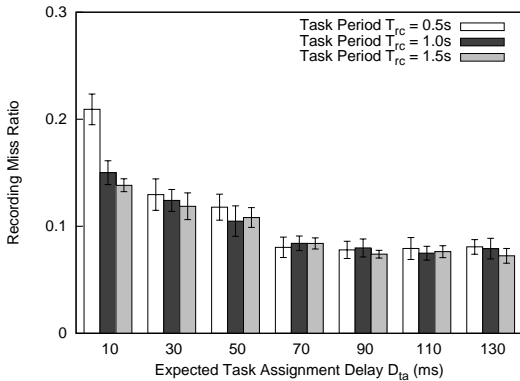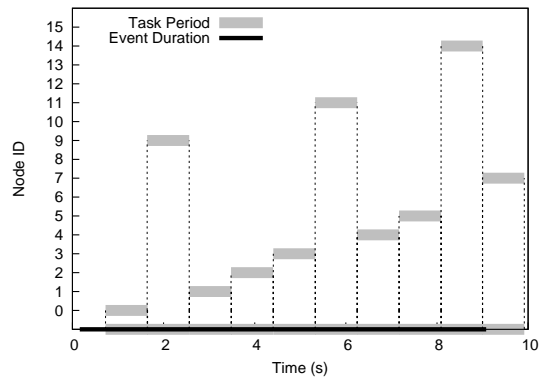
Fig. 6. Recording miss ratio

Fig. 7. One instance of recording a mobile acoustic object

recording miss ratio is due to the initial leader election delay when nodes are forming a group (and no one records). The first leader election, group creation, and first task assignment take 0.7 seconds on average. Divided by the event duration of 9 seconds, it yields an average recording miss ratio of about 0.08. If desired, the initial startup miss can be eliminated by the *prelude* optimization we proposed in Section II-A.1.

Note that when D_{ta} is small ($<=30$ms), a longer task period (T_{rc}) has a lower recording miss ratio. The reason is that for a certain acoustic event, a longer task period requires fewer invocations of the task assignment process, which causes fewer recording gaps. When D_{ta} is large enough, task period has no impact on the recording miss ratio anymore because recording gaps between task assignments are essentially eliminated. The only source of misses comes from the startup delay when the object first enters the network.

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.

Figure 7 plots the recording periods of the nodes, as well as the acoustic event duration of one instance of our experiments in which task period is 1.0s, and estimated task assignment delay is 70ms. Note that not all nodes performed a recording task due to the cooperative task assignment. Recordings happened at different nodes seamlessly. Also note the recording miss at the very beginning when the acoustic target enters the network and nodes are still in the leader election phase.

To better appreciate the efficacy of our cooperative recording subsystem, in Figure 8 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 8(a) plots the sensor readings of the mote held by the person, while Figure 8(b) plots the sensor readings of all the EnviroMic nodes that performed recording tasks, 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`
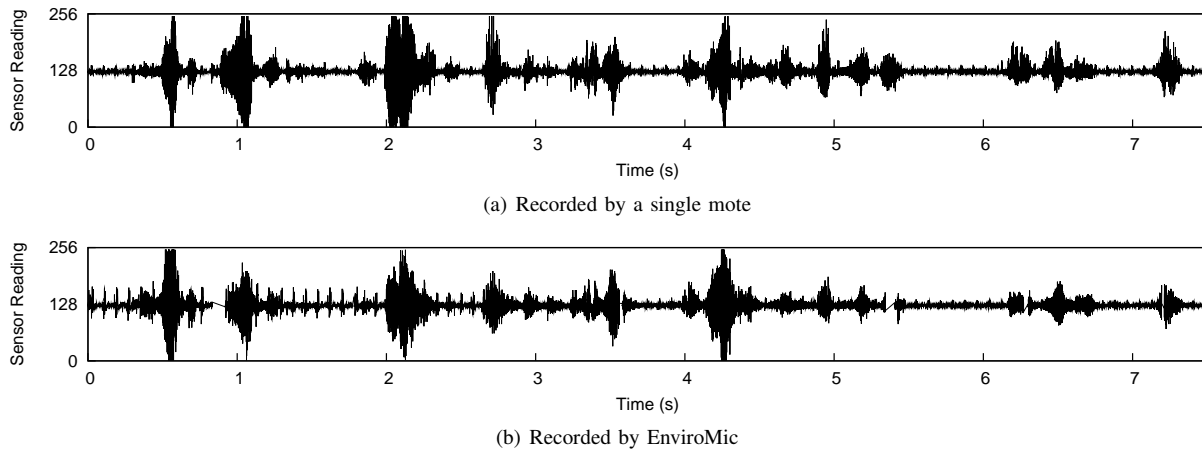
(a) Recorded by a single mote

(b) Recorded by EnviroMic

Fig. 8.   Recording voice of a moving human being

B. Distributed Storage Balancing

In this section, we evaluate the performance of distributed storage balancing. First, we describe the testbed. Next, we present the experiment results, including different performance metrics, such as miss ratio, redundancy ratio and the number of load transfer messages. We also give intuitive illustrations showing spacial distributions of storage occupancy.

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 9). We inject controlled acoustic events into this testbed to achieve repeatability in our experiments, so that we could perform valid comparisons and empirically determine the effects of system parameters.
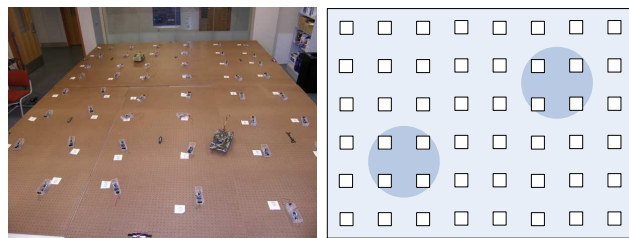

Fig. 9.   Indoor testbed setup

Controlled acoustic events are generated as follows. We use two acoustic sources (laptops) as event generators to play audio clips as events. The locations of the two sources are shown as shaded circles in Figure 9. All events are generated following a Poisson-distributed event arrival process with an expectation of 20 seconds between the start of two consecutive events. The duration of each 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 four 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 β_{max} values in the load balancing subsystem, where we choose values of 2, 3 and 4, respectively. Recall that the actual β value varies linearly between 1 and β_{max}, depending on current TTL. A larger β_{max} means that the load balancing subsystem is less sensitive to load imbalance. We also use two baselines (in which load balancing is disabled). 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), and estimate which mechanism accounts for what part of this improvement.

2) Experiment Results: We now present the results. Two important metrics of load balancing are recording misses and data redundancy. We want recording misses to be low so that recordings can capture the fidelity of acoustic events with minimal data loss. We want data redundancy ratio to be low so that EnviroMic can fully utilize storage capacity of the network.
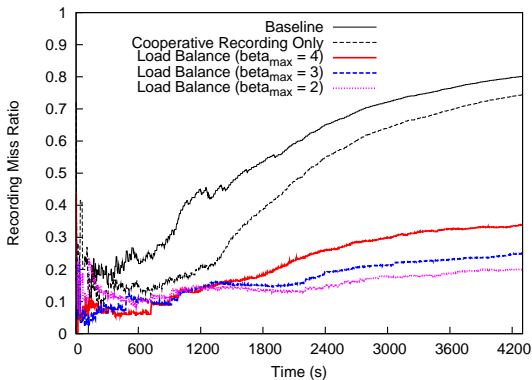

Fig. 10.   Comparison of acoustic recording miss ratio

Recording misses come from two major sources in this experiment. First, as described in Section II, 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 β_{max} is high (i.e., when the load balancing sub-system is less sensitive to data distribution imbalances), one node may not balance its l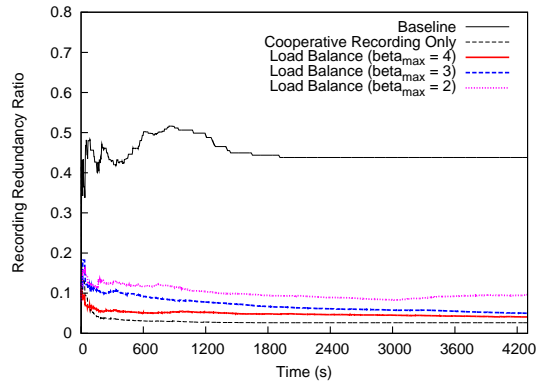oad in time to avoid storage overflow, thereby leading to recording misses, especially when its neighbors are also low on remaining storage space. The results of our experiments on recording misses are shown in Figure 10.

In Figure 10, we compare five curves, including the two baselines. Observe that in Figure 10, TTL-based load balancing achieves a significantly better performance in terms of miss ratio compared to the two baselines. In these baselines, after the four nodes that can detect events fill up their storage spaces, the miss ratio increases considerably. For example, by the end of the experiment, 80% of the data are lost when only local recording is used. On the other hand, 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, four 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.

Note that, 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. There is therefore no point in evaluating the energy cost of load balancing on the current mote prototype.

The second metric we evaluate is the recording redundancy ratio, which we define as the ratio between redundant recordings and all recordings. Intuitively, this metric reflects how efficient we are in eliminating data duplication. We carried out this experiment using the same settings as above, and plot the redundancy ratio in Figure 11. 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, note that when β_{max} is lower, the observed data redundancy ratio is higher. The reason

Fig. 11.   Comparison of acoustic recording redundancy ratio
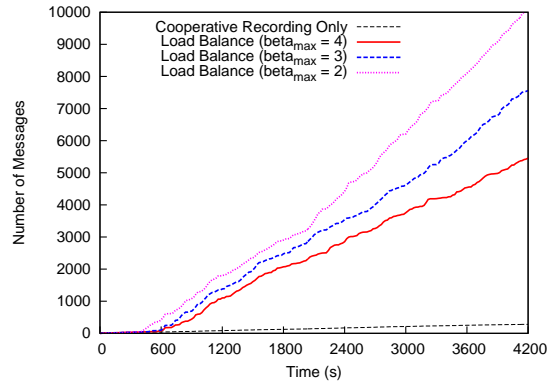

Fig. 12.   Comparison of the number of data transfer messages

is that a lower β_{max} leads to more data transfers. Such transfers may not be completely reliable: one node may replicate its data in multiple neighbors incidentally, resulting in a higher redundancy ratio. 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 or 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.

We also compare the amount of control messages across different settings during the experiment. Control messages include both task assignment messages and load transfer messages. This metric reflects the overhead of load balancing sub-system. The comparison results are shown in Figure 12. In this figure, the baseline without cooperative recording is not included because it does not generate any control messages. For the other four settings, as expected, a β_{max} value of 2 generates the most control messages, because it is the most aggressive to transfer load between nodes. Additionally, observe that the number of control messages increases almost linearly with time for the four settings plotted. This is intuitive because our event injection has a constant rate. Therefore, this observation implies that the incremental overhead for load balancing is proportional to the newly captured data as long as the network capacity is not exhausted.

We now illustrate the spatial distribution of storage occupancy and overhead, measured in the number of control messages sent, from an example run. The evaluation settings are the same as above, except that we do not include the two baselines, since they do not perform load-balancing. We select $\beta_{max} = 2$. For other β_{max} values, the results are similar.

Ideally, at any instant, we hope that acoustic recordings can be distributed evenly in the face of uneven spatial distributions of acoustic sources. We plot contours of spatial data distributions at times $1500s$, $3000s$ and $4500s$, respectively, in Figure 13. First, observe that although the two acoustic sources are relatively far away from each other, data storage is spread out quite evenly in the whole area. Second, observe that the areas near the event sources appear to be the most dense, in terms of storage occupancy, across the three time points. This observation is intuitive given the way the load-balancing mechanism works. A third observation is that in part (c), we notice a *boundary* effect, where nodes in the (very quiet) bottom-left area are pushed a significant amount of data, but have not been able to transfer them out because of the high capacity consumption of nodes surrounding them. Observe that nodes in very quiet areas have a low incoming data rate and thus a very high TTL, causing them to be loaded up with data to a higher level.
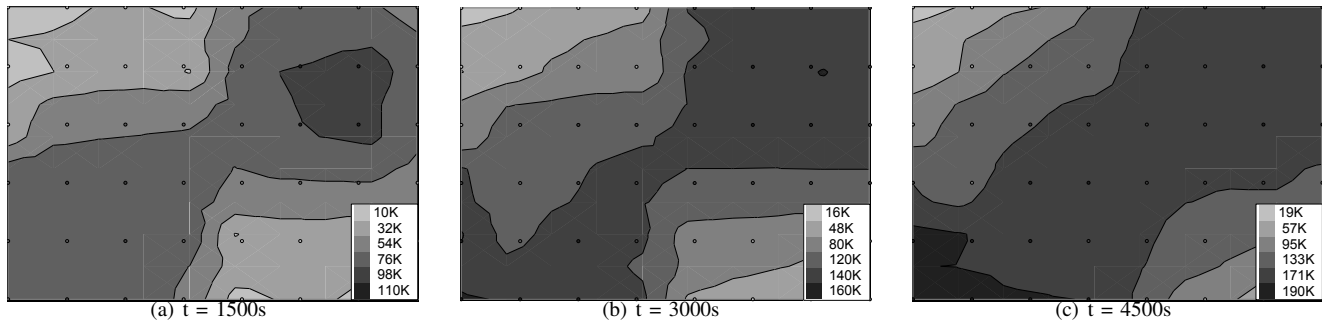
(a) t = 1500s   (b) t = 3000s   (c) t = 4500s

Fig. 13. Spatial distribution of storage occupancy, in bytes, from an example indoor run

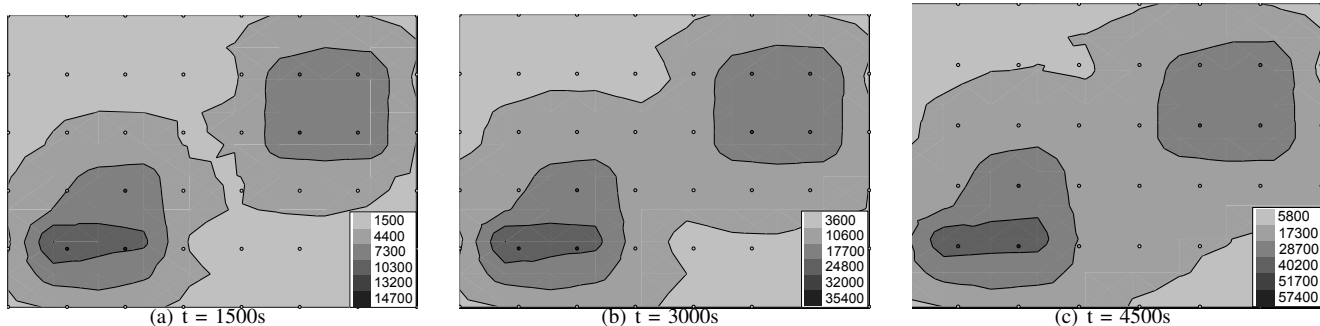

(a) t = 1500s   (b) t = 3000s   (c) t = 4500s

Fig. 14. Spatial distribution of load transfer overhead, in the number of messages, from an example indoor run

Similarly, we plot contours of spatial distributions of the number of control messages sent in Figure 14. Observe that the nodes near event sources generate significantly more messages than the other nodes, because these nodes are expected to record more data, requiring more load balancing transfers to avoid storage overflows. Furthermore, by comparing Figure 14 to Figure 13, we observe that generally, the number of message transfers for a node is correlated with its storage occupancy. The reason is that when nodes record more data, they become more aggressive in transferring these data out, generating a larger number of load-balancing messages.

C. 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 15). On the west side of the forest is a road where vehicles pass by during the day. The experiment was conducted on a day in April 2006. The motes, enclosed in plastic containers, are attached to the

(a) A bird-eye view of the deployment   (b) A mote installed on a tree

Fig. 15. Deployment in a natural forest

trunks of the trees. The deployment area is approximately 105ft×105ft. We were not able to deploy the motes as a grid (to facilitate post-processing of recorded data) because the trees in the forest are in irregular positions. We therefore had to reconstruct the map (Figure 15(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
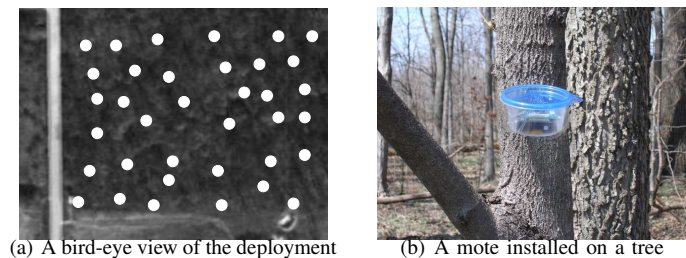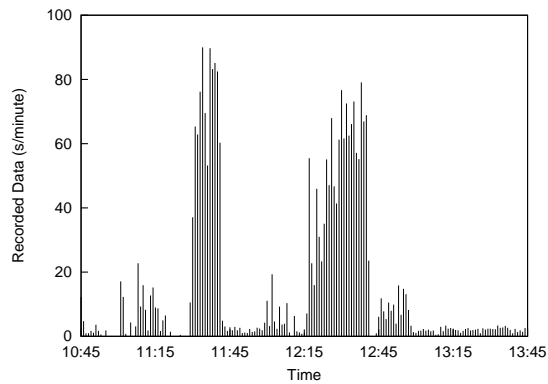
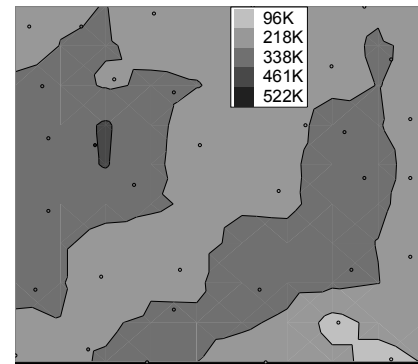Fig. 16. Amount of acoustic event data over time

Fig. 17. Amount of acoustic data, in bytes, generated in different locations

motes in the containers and attaching them to the trees. We collected data recorded by the motes during a period of 3 hours.

The first set of data of interest is how acoustic events are temporally distributed. Figure 16 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 17. 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.

Fig. 18. Distribution of data migrated to nodes for load-balancing

Since we can not repeat the events, we are unable to compare the performance of EnviroMic with baselines as we did in the indoor experiments. However, it is still interesting to see how data recorded in those acoustically-rich regions migrated to other nodes. We therefore picked the node that recorded the highest volume of data, and plotted the amount of data migrated from that node to other nodes in the network in Figure 18. The node that recorded the most data is at coordinate (4, 2) in the rough grid of our deployment. As can be seen in the figure, the node migrated a lot of data to its immediate neighbors, which further migrated some of those data to their neighbors, and so on. This demonstrates the ability of EnviroMic to gracefully handle acoustic event hot-spots.
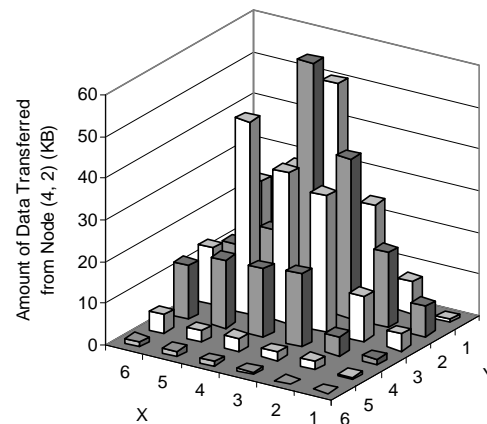
D. Long-term Deployment Plan

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. If it is true, you would expect that EnviroMic would record more vocalizations early in the spring. As males attracted mates they would no longer need to sing at night, so the number of vocalizations would decline throughout the season. Results of the above study, when complete, will be reported separately in a more appropriate forum.

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. One of the limitations researchers face is the ability to collect a sufficient amount of data to begin to determine if different species, individuals in different habitats, or the condition/age of a male affects the intensity with which it sings at dawn. 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.

V. RELATED WORK

1) Acoustic Applications: Sensor network applications have used acoustic sensors for different purposes, including localization [28], surveillance [12], communication [29], and geophysical monitoring [30]. Interestingly, none of these applications let users retrieve raw acoustic sensor samplings: they either use filtered samplings for application needs [12], or use acoustic signals for purposes other than recording [28]. Comparatively, in EnviroMic, we store raw sampling results in a cooperative manner into local storage (flash memory), and retrieve them later upon user request. The way we handle data remotely echoes data-mule [27], 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 [26] 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 [20][17] EnviroMic may characterize the behavior of animals from perspectives totally 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 that may not be possible to obtain without using acoustic sensors. 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.

2) Cooperative Storage: Most storage services were usually designed for individual nodes, such as ELF [5], Matchbox [9], MicroHash [33], and Capsule [22]. Several distributed storage services have also been designed. One example is TSAR [6], which features a two-tier storage and indexing architecture: local storage at the sensor nodes and distributed indexing at the proxies. TSAR is different from our design in that storage is not a cooperative activity among nodes. DIMENSIONS [7] is another system that is designed to store long-term information by constructing summaries at different spatial resolutions using various compression techniques. Our previous work EnviroStore [19] presents a storage service that tries to maximize the effective storage capactiy of disconnected sensor networks, but it focuses on how to take the best advantage of uploading opportunities when data mules [27] become close.

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 [18], where leaders are elected dynamically to coordinate and assign recording tasks to non-leader nodes.

3) Load Balancing: 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 [16], and improving fairness by balancing MAC layer accesses [31]. 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, which can range from web servers to P2P clients, each with a finite resource capacity. The particular resource may be bandwidth, computing power, or storage space. When more resource 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, however, it may be the opposite: 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.

VI. Conclusion

In this paper, we presented EnviroMic, a novel 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 '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 25–38, 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 '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, 2004.

[6] P. Desnoyers, D. Ganesan, and P. Shenoy. Tsar: a two tier sensor storage architecture using interval skip graphs. In *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 39–50, New York, NY, USA, 2005. ACM Press.

[7] D. Ganesan, B. Greenstein, D. Perelyubskiy, D. Estrin, and J. Heidemann. An evaluation of multi-resolution storage for sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 89–102, 2003.

[8] R. K. Ganti, P. Jayachandran, T. F. Abdelzaher, and J. A. Stankovic. Satire: a software architecture for smart attire. In *MobiSys 2006: Proceedings of the 4th international conference on Mobile systems, applications and services*, pages 110–123, 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 '06: Proceedings of the 4th ACM Conference on Embedded Networked Sensor Systems*, November 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 '05: Proceedings of the 3rd international conference on Embedded networked sensor systems*, 2005.

[13] P. Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, 46(2):388–404, 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 '05: Proceedings of the 4th international symposium on Information processing in sensor networks*, page 71, 2005.

[15] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Trans. Netw.*, 11(1):2–16, 2003.

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

[17] T. Liu, C. M. Sadler, P. Zhang, and M. Martonosi. Implementing software on resource-constrained mobile sensors: experiences with impala and zebranet. In *MobiSys '04: Proceedings of the 2nd international conference on Mobile systems, applications, and services*, 2004.

[18] 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.

[19] 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.

[20] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson. Wireless sensor networks for habitat monitoring. In *WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 88–97, New York, NY, USA, 2002. ACM Press.

[21] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi. The flooding time synchronization protocol. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 39–49, New York, NY, USA, 2004. ACM Press.

[22] G. Mathur, P. Desnoyers, D. Ganesan, and P. Shenoy. Capsule: An energy-optimized object storage system for memory-constrained sensor devices. In *SenSys '06: Proceedings of the Fourth ACM Conference on Embedded Networked Sensor Systems*, November 2006.

[23] G. Mathur, P. Desnoyers, D. Ganesan, and P. Shenoy. Ultra-low power data storage for sensor networks. In *IPSN/SPOTS '06: Proceedings of the Fifth International Conference on Information Processing in Sensor Networks*, April 2006.

[24] 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 '05: Proceedings of the Third International Workshop on Measurement, Modeling, and Performance Analysis of Wireless Sensor Networks*, July 2005.

[25] D. Niculescu and B. Nath. Ad hoc positioning system (aps) using aoa. In *INFOCOM '03: the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies*, March 2003.

[26] C. M. Sadler and M. Martonosi. Data compression algorithms for energy-constrained devices in delay tolerant networks. In *SenSys '06: Proceedings of the Fourth ACM Conference on Embedded Networked Sensor Systems*, November 2006.

[27] R. C. Shah, S. Roy, S. Jain, and W. Brunette. Data mules: Modeling a three-tier architecture for sparse sensor networks. In *IEEE Workshop on Sensor Network Protocols and Applications (SNPA)*, 2003.

[28] 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 '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, 2004.

[29] I. Vasilescu, K. Kotay, D. Rus, P. Corke, and M. Dunbabin. Data collection, storage, and retrieval with an underwater sensor network. In *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems*, 2005.

[30] G. Werner-Allen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh. Fidelity and yield in a volcano monitoring sensor network. In *OSDI '06: Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation*, November 2006.

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

[32] 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 '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 13–24, 2004.

[33] 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 '05: Proceedings of the 4th USENIX Conference on File and Storage Technologies*, December 2005.