Design, Implementation, and Evaluation of EnviroMic: A Storage-Centric Audio Sensor Network

LIQIAN LUO
Microsoft Research
QING CAO, CHENGDU HUANG, LILI WANG, and TAREK F. ABDELZAHER
University of Illinois at Urbana-Champaign
JOHN A. STANKOVIC
University of Virginia
and
MICHAEL WARD
University of Illinois at Urbana-Champaign

This article presents the design, implementation, and evaluation of *EnviroMic*, a low-cost experimental prototype of a novel distributed acoustic monitoring, storage, and trace retrieval system designed for disconnected operation. Our intended use of acoustic monitoring is to study animal populations in the wild. Since a permanent connection to the outside world is not assumed and due to the relatively large size of audio traces, the system must optimally exploit available resources such as energy and network storage capacity. Towards that end, we design, prototype, and evaluate distributed algorithms for coordinating acoustic recording tasks, reducing redundancy of data stored by nearby sensors, filtering out silence, and balancing storage utilization in the network. For experimentation purposes, we implement EnviroMic on a TinyOS-based platform and systematically evaluate its performance through both indoor testbed experiments and an outdoor deployment. Results demonstrate up to a four-fold improvement in effective storage capacity of the network compared to uncoordinated recording.

Categories and Subject Descriptors: C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed applications*; H.3.4 [**Information Storage and Retrieval**]: Systems and Software—*Distributed systems*

Preliminary design, implementation, and evaluation results were presented at the 2007 IEEE International Conference on Distributed Computing Systems (ICDCS).
The work reported in this article 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 those of the funding agencies.
Author's address: L. Luo, Microsoft Research, One Microsoft Way, Redmond, WA 98052-6399; email: liqian@microsoft.com.

DOI 10.1145/1525856.1525860 http://doi.acm.org/10.1145/1525856.1525860

General Terms: Design, Experimentation, Performance, Verification

Additional Key Words and Phrases: Sensor networks, applications, acoustics, distributed storage, group management

ACM Reference Format:

Luo, L., Cao, Q., Huang, C., Wang, L., Abdelzaher, T. F., Stankovic, J. A., and Ward, M. 2009. Design, implementation, and evaluation of enviroMic: A storage-centric audio sensor network. ACM Trans. Sensor Netw. 5, 3, Article 22 (May 2009), 35 pages. DOI = 10.1145/1525856.1525860 http://doi.acm.org/10.1145/ 1525856.1525860.

1. INTRODUCTION

This article explores the research challenges in designing resource-constrained acoustic sensor networks that must operate disconnected from the outside world for prolonged intervals of time. Our specific application is to monitor bird vocalizations in a forest in order to study changes in their populations, singing patterns, and spatial distributions in response to various environmental stimuli and habitat disturbances.

Several prior environmental monitoring systems focused on low bandwidth sensors such as light [Batalin et al. 2004], temperature [Mainwaring et al. 2002], motion and magnetic fields [Gu et al. 2005]. Higher bandwidth sensing has been investigated as well, including structural monitoring [Xu et al. 2004], where vibrations in a building were recorded at a frequency of a hundred hertz, and volcano monitoring sensor networks [Werner-Allen et al. 2006], where sensors deployed near an active volcano sampled seismic and acoustic data at 100 Hz. In these systems, the sensors (including special-purpose hardware) were used to sample and report nonscalar data to a base station, leading to challenges in communicating and correlating bursty measurements.

In contrast to these services, where data were uploaded regularly to a base station, this article explores *in-network* storage and energy optimization during *disconnected operation*. Disconnected deployment was addressed in several prior efforts in which data collection occurred only sporadically (e.g., when researchers drove by the field, as in ZebraNet [Liu et al. 2004] or when the monitored targets approached a basestation, as in SATIRE [Ganti et al. 2006]). These applications, however, operated on low bandwidth sensors that did not produce large data volumes. Comparatively, this article explores a new direction where the challenges of high frequency sampling and disconnected deployment coexist in a resource-constrained network.

We utilize acoustic sensing for environmental monitoring. Acoustic sensors are attractive because they have very low energy requirements. Compared to video, acoustic sensing has the added advantage of omnidirectionality and independence from line-of-sight constraints, hence presenting fewer limitations on sensor placement. In the literature, acoustic sensors have been used for various purposes, including localization [Simon et al. 2004; Girod et al. 2006; Ali et al. 2007], surveillance [Gu et al. 2005], communication [Vasilescu et al. 2005], and geophysical monitoring [Werner-Allen et al. 2006]. Interestingly, none of these applications let users retrieve raw acoustic sensor data. They either used data

processed for application needs [Gu et al. 2005], or used raw acoustic signals for purposes other than recording [Simon et al. 2004]. In contrast, we focus on recording and collection of raw acoustic data. The reason is that, at the time of deployment, the range of all possible future uses of the data is usually unknown. Hence, raw data is preferred. For example, the purpose of the current experiment might be to monitor changes in spatial distributions of bird vocalizations over time and correlate them with weather conditions and time of day. However, a subsequent observation of population growth of a predatory species in the area might warrant a second look at the data with the new purpose of extracting interactions (as inferred from acoustic traces) between the bird population and that of the predators. If the original acoustic data are preserved, the new use becomes possible. If the original experiment processed acoustic data immediately, returning only bird population counts, then interaction data with predators are lost. This is an argument for keeping the raw data.

In terms of the architecture of acoustic monitoring systems, the existing solutions include both complex acoustic sensing on high-end processors running Linux [Girod et al. 2006; Ali et al. 2007], acoustic signal processing on low-end devices [Simon et al. 2004; Gu et al. 2005], and hybrid solutions [Hu et al. 2005] utilizing both low-end and high-end devices to collect and analyze acoustic samples respectively. This work falls into the last category. It collects raw acoustic samples while relying on off-line, back-end processing for sophisticated event detection and classification.

The main contribution of EnviroMic is a design that maximizes the amount of acoustic data collected and stored by the resource-constrained sensor network during a prolonged interval of disconnected operation. Equivalently, we minimize data loss due to resource constraints. Data retrieval, in our model, is intermittent. For example, it might be achieved by sending data mules into the field or by physically collecting the sensor nodes after the end of the experiment (e.g., in polar deployments sensors often need to be physically collected before winter season).

The rest of the article is organized as follows. Section 2 describes our model for acoustic, resource-constrained sensor networks that are operating in a disconnected mode. A justification is presented for our particular implementation platform that we use as a proxy to experiment with this class of networks. Section 3 presents the system design. Section 4 presents the details for EnviroMic implementation. Section 5 analyzes evaluation results of EnviroMic based on both indoor and outdoor experiments. Section 6 reviews related work. Section 7 concludes the article.

2. SYSTEM MODEL, CHALLENGES, AND SOLUTION APPROACH

We consider a network of acoustic sensors that monitors a remote bird habitat while operating disconnectedly. A network is needed because the monitored area exceeds the range of a single microphone. This is the case in a forest or on a corridor (a line of trees hospitable to birds separating two fields or lining the sides of a country road).

2.1 Resource Assumptions

Key to our contribution is the notion of system resource constraints. Our design makes three general assumptions on resource constraints in disconnected acoustic networks. We present and justify them below:

—*The storage scarcity assumption*. The primary resource constraint we consider in our network is storage. To justify this decision in the face of decreasing storage cost and increasing volume, consider a high-end microphone sampled to produce CD-quality sound (44.1 kHz with 16 bit samples). The daily uncompressed data volume amounts to more than 7.6 GB/day. Solid-state storage devices, such as flash memory, are preferred for reliability reasons to mechanical hard drives, especially in harsh weather conditions. Unfortunately, such devices run out of storage in a fraction of a day, to a few days, at that rate. Hence, it is important to explore aggressive compression, load sharing, and redundancy elimination methods to improve effective storage capacity.

—*The computing resource scarcity assumption*. Another important resource constraint we assume is energy. For example, solar panels do not work well under the forest canopy. The need for energy savings entails economy in other resources. In particular, acoustic networks operating at high sampling rates cannot benefit much from duty-cycling. The computing device used is permanently on and should thus be a low-power device in order to last for a prolonged deployment. Hence, computing capacity is limited. Observe, for example, that solar panels that can support a laptop for continuous operation (rather than a couple of hours of bright daylight) may cost upwards of $1000. It is therefore interesting to explore architectures that do not require high-end processing capabilities.

—*The communication energy availability assumption*. Finally, we believe that an acoustic recording system will be mostly silent. Indeed, the bulk of communication is needed only to eventually upload data or perform storage load balancing. Radios can be turned off for the rest of the time to save energy. This takes only a negligible portion of the total system deployment time. For example, consider a CD-quality recording system with solid state storage. Using a contemporary WiFi radio (with an effective throughput of 2 Mbps), it takes only about 6.5 minutes to send an entire day of CD-quality recording (0.76 GB when acoustic events occur 10% of the time). Multiplying that time by the ratio of communication device power consumption to the computing device power consumption gives the approximate corresponding reduction in lifetime due to energy spent on communication. Simple calculation shows that the ratio is approximately 2 for both MicaZs and typical embedded PCs.[1] In other words, it costs about 13 minutes of energy lifetime to transfer the

[1]On MicaZs, the ratio of the maximum Tx power consumption of the CC2420 radio to the active power consumption of the ATmega128L processor is 2.175. For embedded PCs, the ratio of the Tx power consumption of a typical 802.11 radio (the Ambicom WL1100C-CF radio card) to a typical embedded processor (the Intel PXA255 processor) is 2.208.

data of one day. This translates to 1% reduction of energy lifetime, which is obviously negligible.

These assumptions motivate every aspect of our system design. To experiment with resource-constrained systems that satisfy the assumptions and facilitate architectural exploration on a laboratory platform, we developed an inexpensive experimental system prototype that is easy to test, maintain, store, and modify. This prototype, called *EnviroMic*, has all the features of a full-fledged acoustic monitoring system, except that it achieves a lower monitoring fidelity. We built our acoustic network prototype using MicaZ motes equipped with MTS300 sensor boards [Crossbow Technology Inc. 2006]. MicaZ motes represent an extreme scaled-down version of resource-constrained acoustic recording networks. Very importantly, they satisfy the three previous assumptions. First, they are storage-constrained, satisfying the storage scarcity assumption. Second, they have very simple micro-controllers, satisfying the computing resource scarcity assumption. Finally, it takes only a few minutes to upload the entire MicaZ flash storage (about 0.5 MB) via the mote's 802.15.4 radio (nominally, 250 Kbps), thus satisfying the communication energy availability assumption. EnviroMic presents the first implementation of a mote-based audio sensor network for recording, storing, and retrieving environmental acoustic traces geared for a prolonged interval of disconnected operation.

It should be noted that even though EnviroMic was built upon mote-based sensor networks, the contributions presented in the work are relevant independently of whether or not motes are used. As aforementioned, even in a scaled-up system that uses using high fidelity microphones to record CD-quality sound into solid state storage on embedded PCs, the assumptions still hold. The proposed solutions therefore remain valid and beneficial.

2.2 Design Challenges and Solution Approach

Networked deployment of large numbers of acoustic sensors presents several challenges 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 important to ensure that EnviroMic limits redundant recording such that storage is used more efficiently. This is achieved by a co-operative recording mechanism, which rotates the task of recording among nodes near the perceived source.[2] The recording service, which implements its own specialized file-system, produces files distributed across neighboring nodes. It attempts to create a single distributed file for each continuous acoustic event.

—Second, since we assume a constrained CPU bandwidth, our design ensures that the recording device (one whose turn to record has come) at any point

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

in time is relieved from other heavy-duty computing or communication responsibilities to focus on high frequency sampling. Hence, we coordinate the recording and other tasks such that they do not interfere, thereby ensuring higher recording quality (e.g., no skips or irregularity).

—Third, the high data volume generated by an acoustic source, coupled with the potentially uneven spatial distributions of such sources, may cause storage on some nodes to overflow while others may still have available storage space. Hence, EnviroMic balances recorded data across nodes to eliminate storage hot-spots and to make use of storage capacity that is not in the direct vicinity of frequent sound sources.

—Finally, a data retrieval subsystem provides a simple mechanism for extracting data from the network, which then can be submitted to a back-end server for sophisticated analysis.

In summary, the major contributions of our work are as follows:

—A cooperative recording protocol that seamlessly distributes the task of recording among nodes, and therefore, achieves a more balanced distribution of the recorded data with little overhead.

—A distributed storage balancing service that eliminates imbalances in storage utilization and prolongs the lifetime of the system as a lossless data collector.

—A prototype implementation that exposes the challenges posed by resource constraints in audio sensor networks.

—Extensive evaluation on both indoor testbeds and outdoor locations that verifies the effectiveness of the proposed algorithms in realistic settings and environments.

Last, observe that while we use MicaZ motes in this article as a reduced-cost prototype and proxy for other systems that satisfy the resource constraint assumptions previously presented, it should be noted that investigating the capabilities of mote-class platforms is not in itself without merit. With advances in NAND flash, new mote prototypes are now available [Mathur et al. 2006b] that interface Mica-class processing and radio hardware to up to 512 MB of flash memory—a three orders of magnitude improvement over MicaZ motes. 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 mote-class hardware in its own right. New generations of motes, such as mPlatform [Lymberopoulos et al. 2007], also have additional higher-end processing cards and radios, allowing for bulk transfer of data at a higher rate when needed. Observe that higher bandwidth radios such as WiFi are much more energy efficient per bit than 802.15.4. Hence, when they are used for bulk transfer and powered down when not in use, such radios will maintain the communication energy availability assumption in future motes (otherwise disrupted by the presence of a larger flash that is too expensive to upload over 802.15.4).

3. SYSTEM DESIGN

The primary concern of EnviroMic is to maximize the effective storage capacity of a sensor network, so as to maximize the amount of useful raw data a scientist can collect about the environment in a single experiment. We assume that back-end base stations perform more sophisticated analysis on data. Hence, we do not do application-specific local processing on sound clips. For example, we do not care to reliably recognize which clips correspond to the same bird. It is the job of the back-end, when data are finally uploaded, to do any needed analysis such as counting bird populations, inferring social communication patterns, or recognizing that two files, refer to the same vocalization. A description of such a back-end appears in Butler et al. [2007]. This article is concerned with the design of the acoustic sensing system that feeds such a back-end and tolerates long periods of disconnected operation. In the following, we describe the main EnviroMic subsystems.

3.1 Cooperative Recording

Recording in EnviroMic is sound activated. While microphones are continuously sensing, nothing is recorded unless it exceeds the long-term running average of background noise by a sufficient margin. Cooperative recording refers to the act of splitting the task of recording an acoustic event among multiple sensors.

For application scenarios requiring more accurate and complete monitoring of the acoustic events, the cooperative recording mechanism can be applied in a different way to reduce redundant recording: (1) when events are present, cooperative recording is turned off to record as much as possible; and (2) when nothing is apparently happening, cooperative recording is activated to provide distributed coverage of non-event periods, which might later be found to contain useful data.

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 of continuous acoustic events. The rest of this section describes the group management and task assignment mechanisms.

3.1.1 *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 ensures that only one copy is recorded. The detailed leader-election algorithm used for this purpose is described in our previous work [Luo et al. 2006]. 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 multi-hop communication. Hence, leaders are associated with local neighborhoods in which the event was heard. Note that the size of such local neighborhoods is determined by a configurable parameter, called *group range*, which should be no greater than the communication range.

Observe that multiple leaders may be elected for the same event, which produces redundant recording. Our design choice is not to guarantee complete elimination of redundancy. Instead, our leader election algorithm simply attempts to eliminate redundancy within up to one-hop communication neighborhoods. This is a compromise between algorithm complexity and performance. With a suitable choice of the group range (e.g., larger than the sensing range for the average acoustic event), redundancy can be eliminated except for very loud acoustic events that are hopefully infrequent.

Observe that, leader election and assignment of the first recording task take some finite time (up to one second in our implementation). Hence, the very beginning of the acoustic event may be 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. This scheme is suitable for low-end platforms, where leader election should not be performed concurrently with recording to ensure that the high frequency recording is not disrupted (e.g., due to communication processing). 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 a continuous sound along its trajectory, EnviroMic attempts to capture this continuity by recording in the same distributed file. EnviroMic features 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 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, for example, in the presence of message loss, or 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. In addition, a temporally separated event (e.g., intermittent vocalizations of the same bird) may also 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 the back-end to extract more accurate information, if needed for the application.

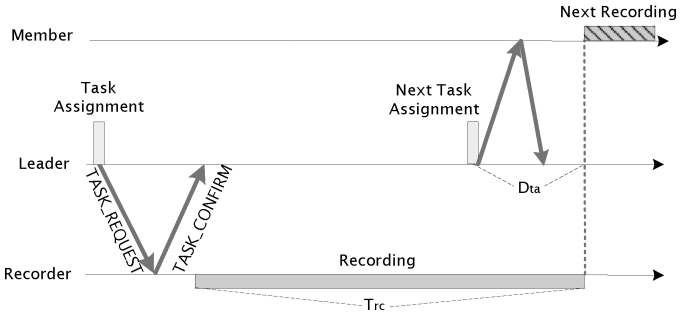


Fig. 1. Task assignment: Delay D_{ta} is necessary for seamless transition between recorders.

3.1.2 *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 3.2) 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 (as depicted in Figure 1). The member echoes the request with a TASK_CONFIRM message, and then starts recording, thereafter called the *recorder*. Recorded data are labeled with the current timestamp, the node ID, and event (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*.

To make the algorithm robust to sparse faulty nodes that keep reporting spurious detections, the group management algorithm exposes a configurable parameter called *minimum detection degree* M. The leader does not start its job (assigning recording tasks) unless there are at least M nodes detecting simultaneously, thus limiting false positives by sporadic faulty nodes. M could be set to be 1 to disable this mechanism.

This algorithm does not attempt to isolate the recording of different concurrent sounds. For example, when two birds sing next to each other, the group management algorithm selects only one leader, which in turn assigns only one recorder for the two birds. As mentioned earlier, this is acceptable because it is the back-end's responsibility to analyze acoustic traces. A more damaging problem is that the algorithm, by eliminating redundancy, can lead to data loss. Since nodes take turns recording parts of an acoustic event, at any given time some nodes are not recording. Those nodes might then miss other local events in their immediate neighborhood. A partial remedy is to have the leader choose multiple members, instead of only one, to simultaneously record their part of an event. However, short of using all nodes all the time, losses may still occur due to the reduced spatial resolution of recording. One can view our

attempt at the elimination of redundancy as an instance of sampling. In general, statistical sampling is a valid technique with which to infer information on populations from only a subset of population measurements. In our specific case, sounds occurring at the same time in the same neighborhood are spatially down-sampled to only one recording. As long as back-end computations (e.g., of bird populations, and their spatial distributions) can account for the nature of the sampling, all is well. For application scenarios where more exact data is needed, the cooperative recording mechanism can be turned off altogether (leaving only the load-balancing mechanism described later). Hence, the user is in control of the chosen trade-off point.

Seamless Transition Between Recorders. In order to achieve a seamless transition from one recording node to another (to ensure that there is no gap between consecutive recording intervals), the leader must initiate a new round of task assignment before the current recorder retires (after T_{rc}). However, task assignment itself does take time. Therefore, we let the leader initiate a new task assignment at $T_{rc} - D_{ta}$ rather than T_{rc}, where D_{ta} is the expected task assignment delay, shown in Figure 1. 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 is 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.

Robustness to Message Loss. To verify the success of task assignment, immediately after sending out the TASK_REQUEST message, the leader starts a timer to expect a TASK_CONFIRM from the recorder. 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 attempt of selecting a new 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 overhearing, shown in Figure 2. 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 or a TASK_CONFIRM message is received by the leader, the leader is assured that the task assignment is done, and hence can schedule

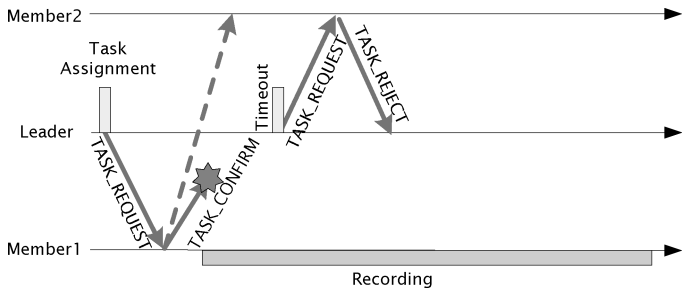


Fig. 2. Task assignment: Reduce the impact of message loss through overhearing.

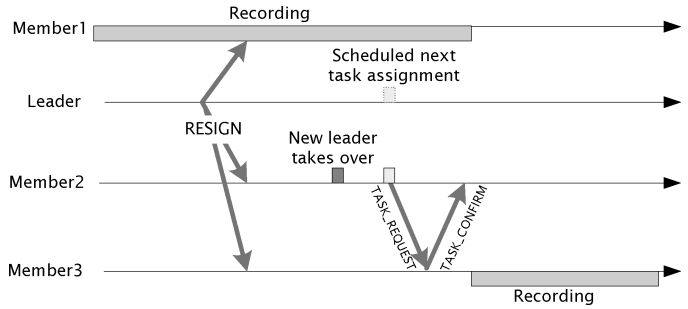


Fig. 3. Task assignment: Leader handoff is unlikely to cause recording misses.

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 2, if the TASK_CONFIRM sent by $Member_1$ was not heard by $Member_2$, both nodes would end up recording simultaneously.

Seamless Leader Handoff. Every node that hears the acoustic event (as opposed to the leader only) maintains a list of nodes that also hear the event (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.

Note that leader handoff is very unlikely to cause recording misses or duplications, although it is not synchronized with recording task rotation. As shown in Figure 3, when the current leader resigns because it cannot 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.

In conclusion, the task assignment algorithm achieves the goal of maintaining only one active recorder at any given time-point by incurring minimum overhead. This is coherent with one of the main design goals of EnviroMic, to reduce data redundancy and therefore prolong storage lifetime. Besides robustness to message loss, the algorithm is also highly resilient to node failures since it dynamically selects recorders from currently alive nodes and the elected nodes serve as recorders only for a short period of time.

3.2 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 start to 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 (1) the time when its flash will saturate at this rate if it does not move data out, and (2) the time when it will run out of energy if it moves data. When an imbalance in storage utilization occurs, a node decides whether or not to move data based on which of the two options maximizes the remaining lifetime. 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, $Exp(R_{event})$ (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 the radio must be active to transmit at rate $R_i(t)$. The load-balancing subsystem 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 $TTL_{storage}$ is the bottleneck, in the rest of this section, we use TTL to refer to $TTL_{storage}$ unless otherwise stated.

Note that thse 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 nor 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 simply extends $TTL_{storage}$ and TTL_{energy} with the same proportion. The bottleneck TTL remains the same.

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 3.1.1 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:

$$\frac{TTL_j}{TTL_i} > \beta_i, \tag{1}$$

and its current TTL_{energy} is larger than its $TTL_{storage}$, it requests 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 $TTL_{storage}$.[3] The parameter β_i determines how sensitive nodes are to storage imbalance. In practice, we set an upper bound β_{max} for β_i, and let β_i vary 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.

Note that, in the case where a node, or a small group of nodes, generates a large amount of data due to defective sensors or noise, the whole network might get filled up by such data due to the load balancing mechanism. To protect the

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

system against such babbling nodes, an additional control knob is adopted. Namely, each node maintains a counter of the total size of data recorded from its own sensor (in appropriately large size units). Node j accepts node i's data migration requests only when the difference between the total amount of local sensor data recorded at node j and that at node i does not exceed a threshold γ (called *maximum expected local imbalance*):

$$\Gamma_j - \Gamma_i < \gamma,$$

where Γ_i is the total amount of data recorded at node i. The introduction of the configurable parameter γ prevents faulty sensors from generating inordinate amounts of data compared to their neighbors, hence drowning the network. A babbling node that exceeds this bound will be cut off by its neighbors, thus limiting the amount of damage it can cause. However, for application scenarios where arbitrarily large differences in detections among neighboring nodes are expected, γ should be set to infinity, which essentially invalidates the protection against babbling nodes.

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.

3.3 Data Retrieval and Analysis

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 then send these chunks and their file IDs along the spanning tree up to the user.

Further interaction with our system user revealed that a single hop version of the aforementioned retrieval scheme was 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.

Finally, analysis of the retrieved data is critical for scientific purposes. In the design of EnviroMic, we adopted a simple acoustic detection algorithm that merely filtered out silence or constant background noise. To distinguish events of interest (bird vocalization) from unwanted events (e.g., sudden wind gusts and extraneous sounds), it is necessary to further apply sophisticated acoustic detection and classification algorithms upon the collected data. A significant literature has addressed this problem adequately. Software toolkits (e.g., Butler et al. [2007]) exist that implement acoustic recognition algorithms. Not to replicate the existing effort but leverage it, we focused on the acoustic collection front-end.

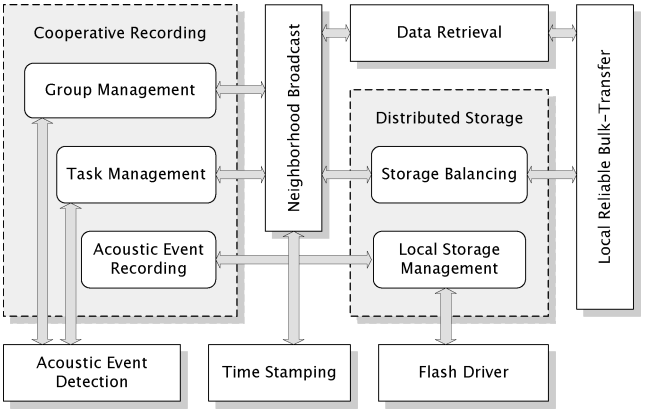


Fig. 4. Modules of EnviroMic implementation.

4. IMPLEMENTATION

4.1 Implementation Overview

To experiment with EnviroMic on resource-constrained hardware platforms conforming to our assumptions, we implemented a prototype of EnviroMic on MicaZ motes running TinyOS. We think of MicaZs as an extreme scaled-down version of resource-constrained acoustic recording networks. They are also good representatives of a large class of off-the-shelf hardware platforms with various sensing and storage capabilities.

The implementation consists of 12 nesC modules, and 10,282 lines of nesC code. The system occupies 61.5 KB of code memory and 2.8 KB of data memory on MicaZ. Figure 4 gives an overview of the major modules of our implementation, and the interface relationships between them. A few words are in order to describe some modules in Figure 4 that are not discussed in previous sections.

We chose to use a simple power-based scheme to implement acoustic event detection rather than alternative frequency-based schemes. The reason is that frequency spectrum analysis by FFT is too expensive for the Atmel ATMega 128L micro-controllers on MicaZ platforms, which are low-bandwidth (8 MHz) and do not have native floating-point computation support. The acoustic event detection module maintains two exponentially weighted moving averages for acoustic samples, one having a long averaging interval (to estimate background noise) and the other having a shorter one (to sense deviations). When the difference between the two averages exceeds an empirically determined threshold, a detection event is signaled to upper layer modules.

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 [Maroti et al. 2004]. 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 4. 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 for 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. To initiate load-balancing, an overloaded node sends out a request message. At the reception of the request message, each underloaded node starts a timer whose expiration time is inversely proportional to its available storage capacity. The node that expires first then responds to the overloaded node via a confirm message and suppresses other underloaded nodes. Complete data chunks (256 B each) are then transferred between the two nodes. Upon the reception of a complete data chunk, the receiver commits the data into flash and acknowledges the sender, which then deletes the corresponding data chunk from its own flash. Due to the unreliability of wireless links, it is possible that such acknowledgement messages get lost, thus leading to certain degree of data redundancy.

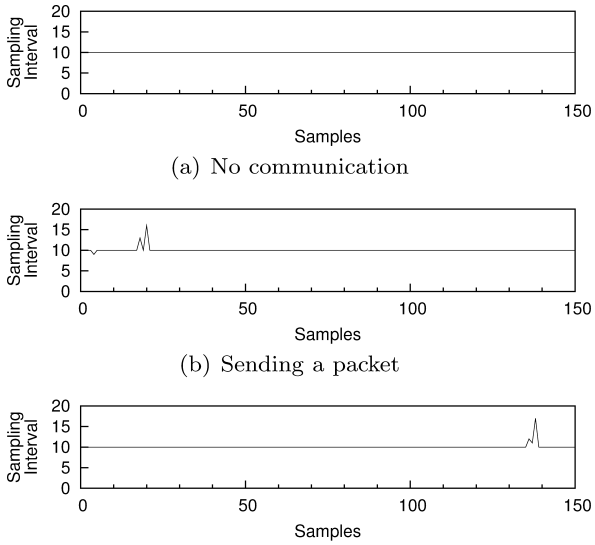
4.2 Implementation Issues

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

4.2.1 *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 cannot 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 disturbances from receiving packets for it can significantly slow down the acoustic sampling as well.

Figure 5 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 5 shows the observed intervals for different scenarios.

In Figure 5(a), the node is exclusively sampling data. Therefore, the observed sampling intervals are fixed at 10 jiffies. In Figure 5(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 5(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. We expect that



(a) No communication

(b) Sending a packet

(c) Receiving a packet

Fig. 5. Measured sampling interval between consecutive samples.

in a higher quality prototype, both the hardware and the acoustic sampling frequency become faster. Hence, the problem may remain on a broad range of embedded processors (without hardware sampling support).

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

4.2.2 *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 (depicted in Figure 6) is associated with certain metadata, including start and end timestamps, a location-stamp (or the ID of the recording

StartTime (4B)	EndTime (4B)	NodeID (2B)	EventID (2B)	DataSize (1B)	Data (243B)

Fig. 6. Structure of a data chunk in local storage.

node, which can be translated to location in a data post-processing stage), and an event (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 base station 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.

5. EVALUATION

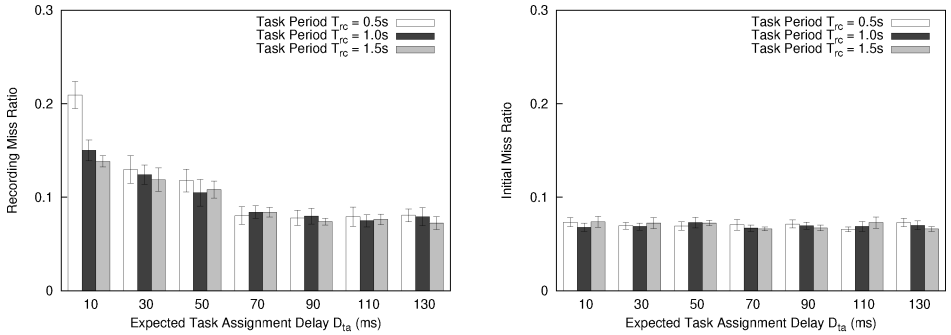
We evaluate EnviroMic using both an indoor testbed and an outdoor deployment in a forest. The acoustic sampling frequency is set to be 2.730 kHz throughout the experiments. The indoor testbed consists of 48 MicaZ motes placed as a 8×6 grid with unit grid length 2 ft. We use this testbed together with controlled acoustic events, described in the following 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.

5.1 Cooperative Recording

Efficiency of the cooperative recording subsystem comprises two related properties. First, we want the acoustic event recording to be complete (no recording gaps). Second, we want to reduce recording redundancy. Our design and implementation of seamless task assignment (Section 3.1.2) ensure 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 3.1.2, 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 3.1.2, 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 short task period means more task assignments for the same amount of acoustic events, leading to high control overhead. On the other hand, a large task period may not work for mobile acoustic events. When an acoustic event is present, the node performing the



(a) Total miss ratio (b) Initial miss ratio

Fig. 7. Miss ratios.

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 7(a) 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 70 ms, and stabilizes at about 8%. Further investigation reveals that this fixed recording miss ratio is mainly 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 about 0.6 seconds on average. Divided by the event duration of 9 seconds, it yields an average initial recording miss ratio of 7% or so as is depicted in Figure 7(b). Therefore, in a long recording, the miss ratio caused by glitches in the recording is about 1%.

Note that when D_{ta} is small ($\leq$30ms), 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, the 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.

Each task assignment incurs certain overhead. Therefore longer task periods are more preferable than shorter ones. However, due to the mobility of acoustic sources, that are too long task periods, result in degraded recording quality (as the source moves further away from the recorder). In the experiment, we consider 1.0 s as a good value for the task period, since (1) it incurs less assignment

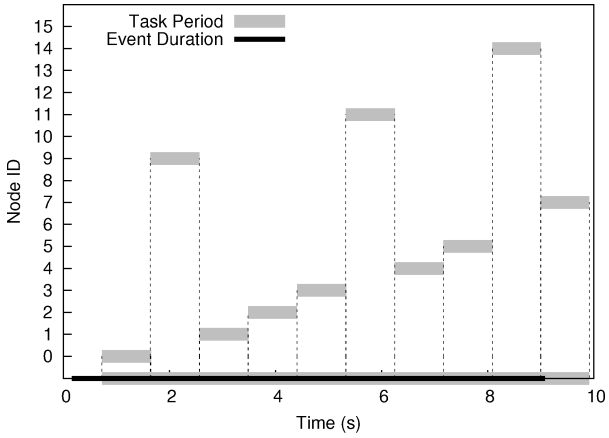
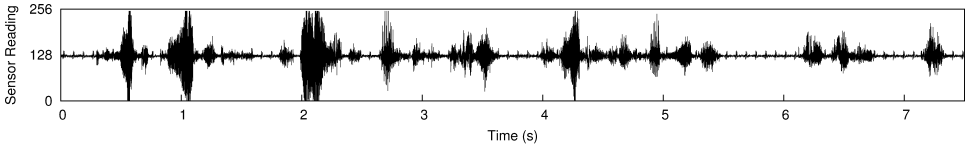
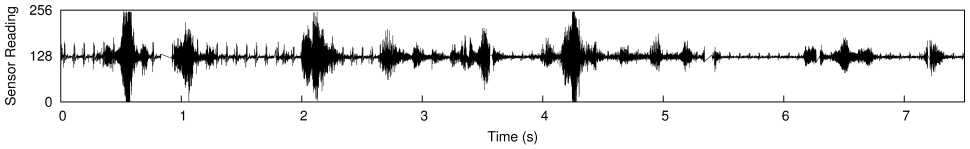


Fig. 8. One instance of recording a mobile acoustic object.



(a) Recorded by a single mote



(b) Recorded by EnviroMic

Fig. 9. Recorded voice of a moving human being.

overhead compared with 0.5 s, and (2) we found that the recorded data quality for a task period of 1.5 s is noticeably worse than that for a period of 1.0 s. Hence, we picked 1.0 s as the task period for the rest of our experiments and accordingly 70 ms as the expected task assignment delay.

Figure 8 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 1.0 s, and estimated task assignment delay is 70 ms. 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 9 we present an experiment of recording human voice. In this experiment, a person read out the title of this article 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 continuously. Its recording

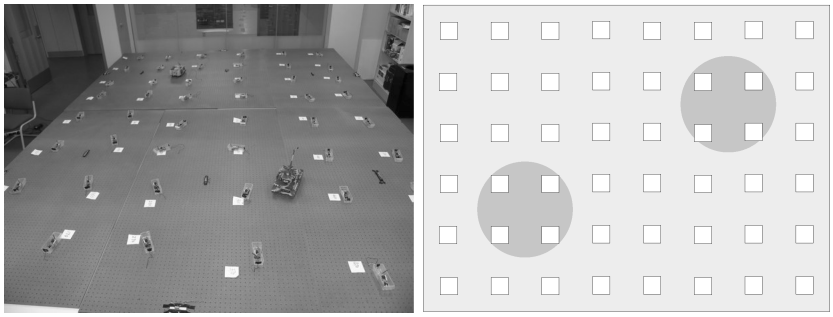


Fig. 10. Indoor testbed setup.

serves as a reference "ground truth", representing the best quality recording of the moving person for the chosen sampling rate (2.730 kHz) and hardware platforms (micaZ motes). If we are successful, we expect to observe that the output of the cooperative recording subsystem is comparable with that of the single mote. Figure 9(a) plots the sensor readings of the mote held by the person, while Figure 9(b) plots the sensor readings of all the EnviroMic nodes that performed recording tasks, stitched together based on their time stamps. The visual similarity of the two figures is obvious. Note that there are small recording gaps, as observed in Figure 9(b). They are caused by the dynamics in task assignment delays. When such delays are longer than 70ms (D_{ta}), we observe recording gaps.

Compared with human voice, birds have a much simpler vocabulary. We also used EnviroMic to record primitive sound, in particular, calls of woodpeckers. By comparing the original digitized sound of woodpeckers to the mote-based recording, it is clear that the mote recording remains quite recognizable. For readers who are interested, all the sound clips of human voice, calls of woodpeckers, as well as other primitive sounds, are available online for qualitative comparison.[4]

5.2 Distributed Storage Balancing

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

5.2.1 *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 10). We inject controlled acoustic events into this testbed to achieve repeatability in our experiments, so that we can perform valid comparisons and empirically determine the effects of system parameters. Controlled acoustic events are generated as follows. We use two acoustic sources (laptops)

[4]`http://research.microsoft.com/~liqian/enviromic/`

as event generators to play audio clips as events. The locations of the two sources are shown as shaded circles in Figure 10. 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 (25% of the length of the experiment). To experiment with load balancing, we allow only four nodes to 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.

5.2.2 *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 the data redundancy ratio to be low so that EnviroMic can fully utilize the storage capacity of the network.

Recording misses come from two major sources in this experiment. First, as described in Section 3, 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 (when the load balancing subsystem 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 11.

In Figure 11, we compare five curves, including the two baselines. Observe that in Figure 11, 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 smallest 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 four-fold

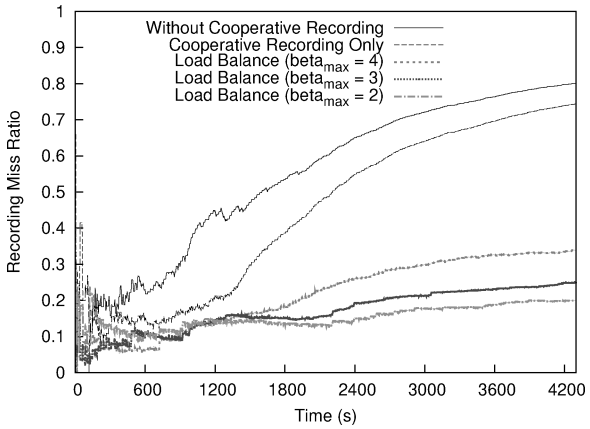


Fig. 11. Comparisons of acoustic recording miss ratio.

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.

According to the communication energy availability assumption, the load balancing energy is almost negligible. Next, we present a back-of-the-envelope calculation on MicaZ motes to validate this conclusion. Let's assume a busy event model where on average, a node records one second of data (2730 B under the 2.73 kHz sampling rate) for every 10 seconds. The storage consumption of one hour would be approximately 960 KB. On the other hand, if we use one hour to transfer, the amount of storage saved would be around 22,500 KB, assuming the effective bandwidth of the CC2420 radio is only one-fifth of its maximum bandwidth 250 Kbps. The calculation tells us sending one hour saves around 23 hours (dividing 22,500 KB by 960 KB) of storage life. Note that it is true sending messages takes more energy than normal operation. Given that the energy cost of the micro-controller on MicaZ motes is about 0.024 J per second and the extra energy cost of the CC2420 radio on MicaZ motes at TX mode is 0.052 J per second, sending one hour actually kills approximately three hours of energy life, which is still relatively small compared with the 23-hour saving in storage life. Therefore, the lifetime reduction due to load balancing is negligible compared with the savings in storage life. For all practical purposes, it can be ignored. There is therefore no point in evaluating the energy cost of load balancing.

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 previously, and plot the redundancy ratio in Figure 12. 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

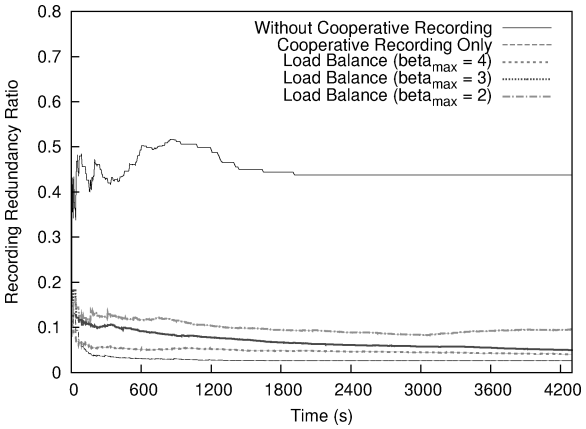


Fig. 12. Comparisons of acoustic recording redundancy ratio.

strategy to reduce recording redundancy. Second, note that when β_{max} is lower, the observed data redundancy ratio is higher. The reason is that a lower β_{max} leads to more data transfers. Such transfers may not be completely reliable: one node may incidentally replicate its data in multiple neighbors, resulting in a higher redundancy ratio. 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 reliably detect the event. 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 quantity 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 cooperative recording and load balancing subsystems. The comparison results are shown in Figure 13. Note that in EnviroMic there is no multi-hop communication—all the messages are one-hop messages. Therefore, the number of messages presents the energy cost of radio communication well. 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 in transferring 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 before, except that we do not include the

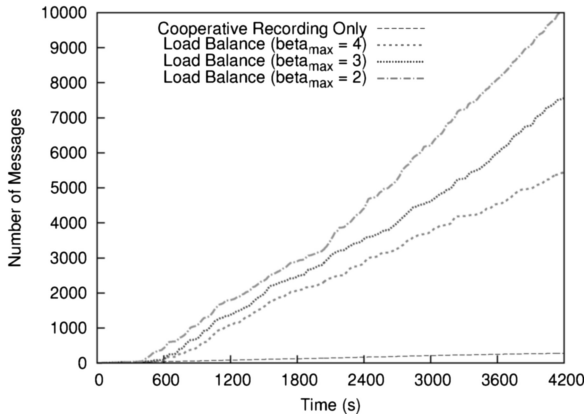
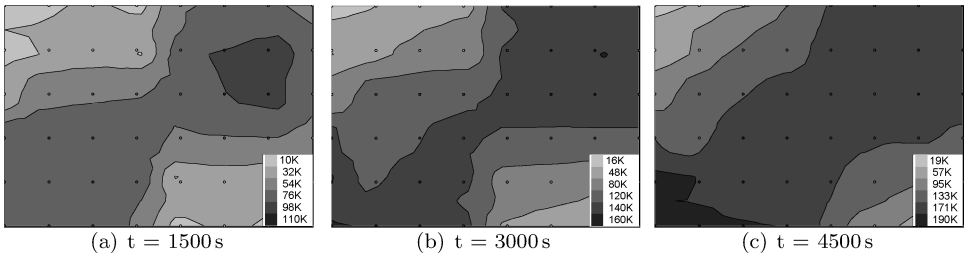


Fig. 13. Comparisons of the number of control messages.



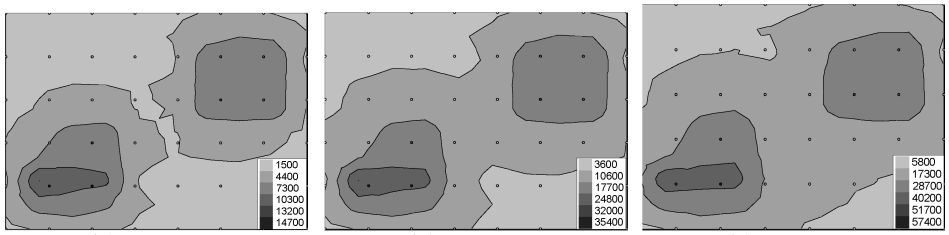
(a) t = 1500 s (b) t = 3000 s (c) t = 4500 s

Fig. 14. Spatial distribution of storage occupancy, in bytes, from an example indoor run.

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 1500 s, 3000 s, and 4500 s, respectively, in Figure 14. Note that the results correspond to the setup in Figure 10. 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 densest, 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.

Similarly, we plot contours of spatial distributions of the number of control messages sent in Figure 15. Observe that the nodes near event sources generate significantly more messages than the other nodes, because these nodes



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

Fig. 15. Spatial distribution of load transfer overhead, in the number of messages, from an example indoor run.

are expected to record more data, requiring more load-balancing transfers to avoid storage overflows. Furthermore, by comparing Figure 15 to Figure 14, 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.

5.3 Preliminary Outdoor Deployment

To understand the efficacy and limitations of our current design and implementation of EnviroMic in more realistic environments, we did a preliminary outdoor deployment of an EnviroMic system based on MicaZ platforms. The purpose of the deployment was to monitor the spatial distribution of birds and their vocalization frequencies. To test the feasibility of using MicaZ motes in achieving that purpose, we first investigated the capabilities of our system indirectly, by answering three related questions, namely: (1) Can the recording of MicaZ motes be used to recognize bird species? (2) What is the range at which MicaZ motes reliably detect and record the sound of a bird? and, (3) How accurately can a sound source be localized? Correspondingly, we did three micro-benchmarks.

In the first micro-benchmark, we recorded sounds of woodpeckers using a group of 16 MicaZ motes. Both the high-fidelity recording and the mote-based recording are shared.[5] The results provide strong qualitative evidence that some bird species are distinctly recognizable from mote recordings.

In the second micro-benchmark, we attempt to validate how effective the system is at detecting bird-cry events. During the experiment, we placed a controllable sound source and a MicaZ mote running the detection module of EnviroMic at difference distances, and then counted—out of the 20 acoustic events we generated at each distance—how many times the mote reported a detection. Figure 16 depicts the relation between distance and the number of detections reported. As we can see, a loud bird can be detected reliably up to 30 ft with MicaZ motes, which means we can deploy motes up to 60 ft apart without sacrificing coverage.

[5]`http://research.microsoft.com/~liqian/enviromic`.

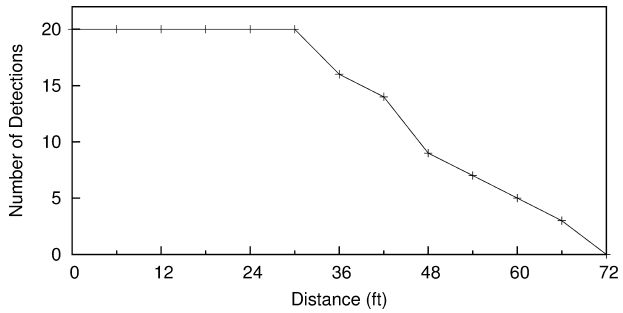


Fig. 16. Number of detections for 20 acoustic events at difference distances.

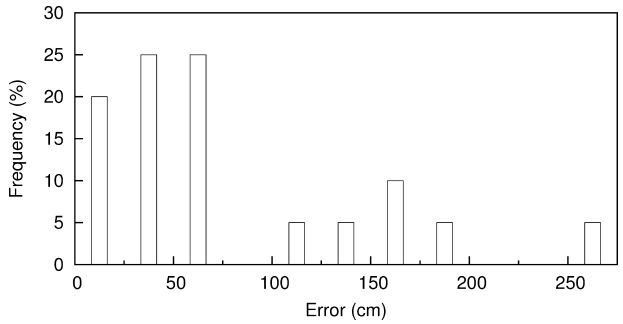
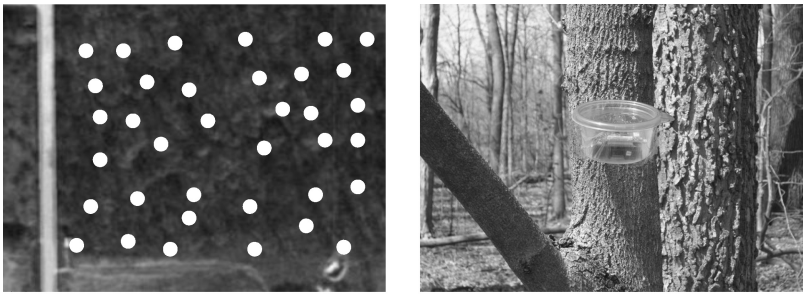


Fig. 17. Distribution of localization errors.

The last micro-benchmark investigates the localization accuracy. We deployed 8 MicaZ motes in a line. The distance between any two neighbors is 5 meters. The rationale for a single-dimensional deployment was to emulate corridor habitats, such as a line of trees separating two fields or stretching along a roadside. Those trees host birds and represent a unidimensional habitat. During the experiment, we placed a controllable source source at a certain location along the line and repeated the experiment for 40 times and for different locations. Based on the recordings we collected from different motes, we calculated the location of the source as a weighted average of the locations of the detecting motes (using their recording length as the weight). Figure 17 depicts the distribution of localization errors (in centimeters). As is seen, 95% of the errors were below 2 meters (in a deployment configuration where sensors were 5 meters apart), which is reasonably accurate.

The results from these micro-benchmarks show that (1) MicaZ motes are able to record recognizable bird sounds, and (2) with a reasonable spacing, they are able to reliably detect such sound sources and accurately localize them.

In the actual outdoor experiment, we deployed a network of 36 MicaZ motes in a forest (Figure 18). On the west side of the forest is a road where vehicles pass during the day. The experiment was conducted on a day in April 2006. The motes, enclosed in plastic containers, were attached to the trunks of the trees. The deployment area was approximately 105 ft$\times$105 ft. We were not able to deploy the motes as a grid (to facilitate post-processing of recorded data)



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

Fig. 18. Deployment in a natural forest.

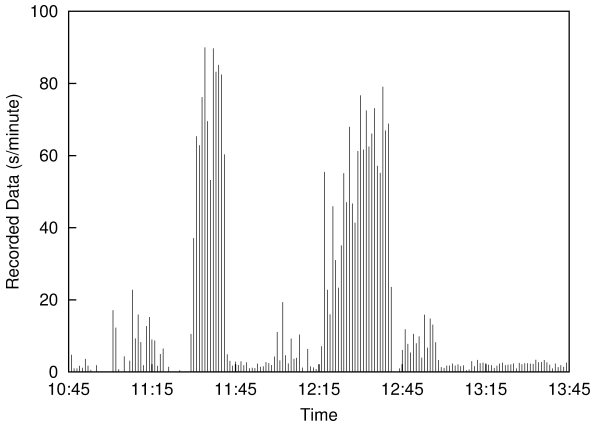


Fig. 19. Amount of acoustic event data over time.

because the trees in the forest are in irregular positions. We therefore had to reconstruct the map (Figure 18 (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 was 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.

The first set of data of interest is how acoustic events are temporally distributed. Figure 19 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 looked at the geographic distribution of the events. From the data stored in the motes, we reconstructed the information on how much acoustic

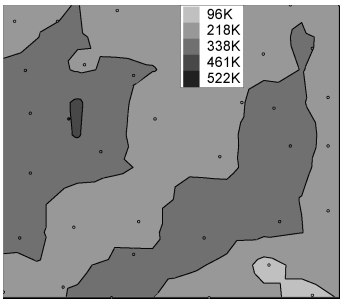


Fig. 20. Amount of acoustic data, in bytes, generated in different locations.

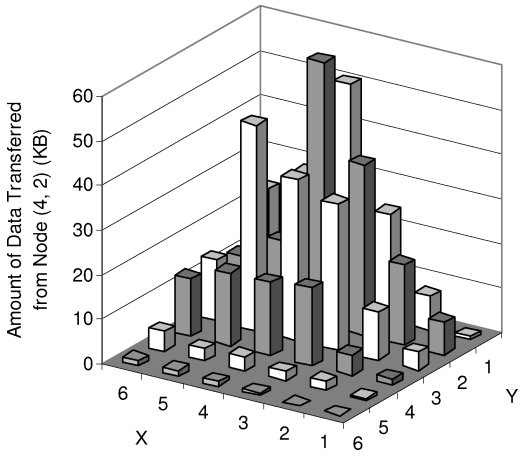


Fig. 21. Distribution of data migrated to nodes for load-balancing.

data was generated, from which sensors, throughout the experimentation period. Mapping node IDs to geographic locations, we plotted a contour graph of the total volume of acoustic events (in bytes) shown in Figure 20. We can see that there are two high data volume regions. The one on the left side was caused by vehicles passing on the road to the west of the forest. The other region rich in acoustic events in the figure roughly matches a trail in the forest.

Since we cannot 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 21. The node that recorded the most data is at coordinates (4, 2) in the rough grid of our deployment. As can be seen in the figure, the node offloaded a lot of data to its immediate neighbors, which further offloaded some of those data to their neighbors, and so on. This demonstrates the ability of EnviroMic to gracefully handle acoustic event hot-spots.

6. RELATED WORK

6.1 Acoustic Applications

Sensor network applications have used acoustic sensors for different purposes, including localization [Simon et al. 2004; Girod et al. 2006; Ali et al. 2007], surveillance [Gu et al. 2005], communication [Vasilescu et al. 2005], and geophysical monitoring [Werner-Allen et al. 2006]. Interestingly, none of these applications let users retrieve raw acoustic sensor samplings: they either use filtered samplings for application needs [Gu et al. 2005], or use acoustic signals for purposes other than recording [Simon et al. 2004]. 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 [Shah et al. 2003], which also uses a store-and-fetch model.

In terms of system architecture, the existing solutions include both complex acoustic sensing on high-end processors running Linux [Girod et al. 2006; Ali et al. 2007], acoustic signal processing on low-end devices [Simon et al. 2004; Gu et al. 2005], and hybrid solutions [Hu et al. 2005] utilizing both low-end and high-end devices to collect and analyze acoustic samples respectively. EnviroMic falls into the third category. It merely collects acoustic samples through MicaZ motes without applying any further signal processing in the network. One of the systems that bears the greatest similarity to our system in terms of architecture is the cane-toad monitoring system presented in Hu et al. [2005]. In the cane-toad monitoring system, a hybrid network of Mica2s (8 MHz processors) and Stargates (400 MHz processors) was used to detect the presence of cane toads of different species. The Mica2s merely collect samples and compress noise before sending them to the Stargates, which then execute FFT and machine learning algorithms to determine the existence of certain cane toad species. This is similar with EnviroMic, which relies on a network of MicaZs for acoustic sampling and base stations for sophisticated analysis. However, since the Mica2 nodes in the cane-toad monitoring system are always connected to powerful Stargates, they upload acoustic samples to Stargates for real-time processing rather than retaining the data in the network. The ultimate outputs of the system are detection reports, which are not storage-intensive. Comparatively, EnviroMic stores all the raw acoustic samples in the mote network for later retrieval. This is especially beneficial for application scenarios where the range of all possible future uses of the data is unknown at the time of deployment. However, they assume the existence of more storage-rich devices for real-time data uploading.

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 focuses on reducing data redundancy. Obviously, other techniques including in-network filters [Greenstein et al. 2006] and data compression algorithms [Sadler and Martonosi 2006; Barr and Asanović 2003; Puthenpurayil et al. 2007] can be easily integrated into EnviroMic to further reduce the data volume to be stored in the network. During the selection of

appropriate filtering or compression algorithms, the trade-offs between compression and energy consumption need to be well considered. Off-the-shelf algorithms should be tailored to fit into the special energy and memory constraints of sensor nodes. Such issues have been well addressed in the literature, and thus are not the focus of our article. To integrate compression techniques into EnviroMic, a compression layer can be inserted between the acoustic event recording module and the local storage management module, so that each data chunk is compressed before being written into the local storage. Multiple compressed data chunks then can be packed together to save storage space. During storage balancing, such compressed data chunks rather than the original data chunks are exchanged between nodes. Note that an on-line decompression module is not required since data can be decompressed after final collection.

Also, EnviroMic has a great potential to be applied in applications that previously did not use acoustic sensors. For example, in animal monitoring [Mainwaring et al. 2002; Liu et al. 2004] 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.

This article is an extension of our previous work [Luo et al. 2007a, b], which focuses more on the design challenges in acoustic sensor networks. Different from the previous work, this article covers in-depth implementation details and presents thorough and comprehensive experimentation results.

6.2 Cooperative Storage

Most storage services were usually designed for individual nodes, such as ELF [Dai et al. 2004], Matchbox [Gay 2003], MicroHash [Zeinalipour-Yazti et al. 2005], and Capsule [Mathur et al. 2006a]. Several distributed storage services have also been designed [Desnoyers et al. 2005; Ganesan et al. 2003; Tilak and Abu-Ghazaleh 2005; Selavo et al. 2007]. One example is TSAR [Desnoyers et al. 2005], 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 [Ganesan et al. 2003] is another system that is designed to store long-term information by constructing summaries at different spatial resolutions using various compression techniques. In LUSTER[Selavo et al. 2007], all the sensor nodes send data to a base station in real time. To enhance reliability, LUSTER utilizes dedicated storage nodes to overhear and store data into local flash. Our previous work EnviroStore [Luo et al. 2007b] presents a storage service that tries to maximize the effective storage capacity of disconnected sensor networks, but it focuses on how to take the best advantage of uploading opportunities when data mules [Shah et al. 2003] become close.

However, none of these storage services is appropriate for EnviroMic because of its unique challenge of achieving cooperative storage. To achieve this purpose,

we use a leader election algorithm similar to those proposed in EnviroTrack [Luo et al. 2006] and State-centric programming [Liu et al. 2003a, b], where leaders are elected dynamically to coordinate and assign recording tasks to non-leader nodes.

6.3 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 [Li et al. 2001], and improving fairness by balancing MAC layer accesses [Woo and Culler 2001]. 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 cannot 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 cannot 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 [Cardellini et al. 1999] and P2P networks [Godfrey et al. 2004]. 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 resources than desired are 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 [Cardellini et al. 1999; Godfrey et al. 2004]. 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.

7. CONCLUSION

In this article, we presented EnviroMic, a low cost prototype of a distributed acoustic monitoring, storage, and trace-retrieval system. The long term disconnected service model for our target applications calls for a design that stores recorded acoustic data in the network until it can be uploaded. 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. Recorded data chunks 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. With the experience gained from the low-cost prototype reported herein, our next step is to deploy a scaled-up solar-powered acoustic monitoring network of high-fidelity microphones attached to embedded PCs to perform CD-quality recording. The new system also satisfies the three resource assumptions made in this article, hence using the same mechanisms ported to the new platform. Future work of the authors will address issues of controlled redundancy, optimization trade-offs for renewable energy, as well as resilience to failures, self-diagnosis, and self-healing capabilities in the remotely deployed system. Also of interest will be combining low bandwidth real-time communication (e.g., querying for summary data or monitoring system health) with high bandwidth intermittent connections (e.g., eventual raw data collection in the field).

REFERENCES

ALI, A. M., YAO, K., COLLIER, T. C., TAYLOR, C. E., BLUMSTEIN, D. T., AND GIROD, L. 2007. An empirical study of collaborative acoustic source localization. In *Proceedings of the 6th International Conference on Information Processing in Sensor Networks (IPSN)*. 41–50.

BARR, K. AND ASANOVIĆ, K. 2003. Energy aware lossless data compression. In *Proceedings of the 1st International Conference on Mobile Systems, Applications and Services (MobiSys)*. 231–244.

BATALIN, M. A., RAHIMI, M., YU, Y., LIU, D., KANSAL, A., SUKHATME, G. S., KAISER, W. J., HANSEN, M., POTTIE, G. J., SRIVASTAVA, M., AND ESTRIN, D. 2004. Call and response: experiments in sampling the environment. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys)*. 25–38.

BUTLER, R., SERVILLA, M., GAGE, S., BASNEY, J., WELCH, V., BAKER, B., FLEURY, T., DUDA, P., GEHRIG, D., BLETZINGER, M., TAO, J., AND FREEMON, D. M. 2007. Cyberinfrastructure for the analysis of ecological acoustic sensor data: a use case study in grid deployment. *Cluster Comput. 10,* 3, 301–310.

CARDELLINI, V., COLAJANNI, M., AND YU, P. S. 1999. Dynamic load balancing on web-server systems. In *IEEE Int. Comput. 3*, 3, 28–39.

CROSSBOW TECHNOLOGY INC. 2006. MTS300 Multi Sensor Board. `http://www.xbow.com`.

DAI, H., NEUFELD, M., AND HAN, R. 2004. Elf: an efficient log-structured flash file system for micro sensor nodes. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys)*.

DESNOYERS, P., GANESAN, D., AND SHENOY, P. 2005. Tsar: a two tier sensor storage architecture using interval skip graphs. In *Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems (SenSys)*. 39–50.

GANESAN, D., GREENSTEIN, B., PERELYUBSKIY, D., ESTRIN, D., AND HEIDEMANN, J. 2003. An evaluation of multi-resolution storage for sensor networks. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys)*. 89–102.

GANTI, R. K., JAYACHANDRAN, P., ABDELZAHER, T. F., AND STANKOVIC, J. A. 2006. Satire: a software architecture for smart attire. In *Proceedings of the 4th International Conference on Mobile Systems, Applications and Services (MobiSys)*. 110–123.

GAY, D. 2003. Matchbox. `http://www.tinyos.net/tinyos-1.x/doc/matchbox.pdf`.

GIROD, L., LUKAC, M., TRIFA, V., AND ESTRIN, D. 2006. The design and implementation of a self-calibrating distributed acoustic sensing platform. In *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems (SenSys)*. 71–84.

GODFREY, B., LAKSHMINARAYANAN, K., SURANA, S., KARP, R., AND STOICA, I. 2004. Load balancing in dynamic structured p2p systems. In *Proceedings of the Conference on Computer Communications (INFOCOM)*.

Greenstein, B., Mar, C., Pesterev, A., Farshchi, S., Kohler, E., Judy, J., and Estrin, D. 2006. Capturing high-frequency phenomena using a bandwidth-limited sensor network. In *Proceedings of the 4th ACM Conference on Embedded Networked Sensor Systems (SenSys)*.

Gu, L., Jia, D., Vicaire, P., Yan, T., Luo, L., Tirumala, A., Cao, Q., He, T., Stankovic, J. A., Abdelzaher, T., and Krogh, B. H. 2005. Lightweight detection and classification for wireless sensor networks in realistic environments. In *Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems (SenSys)*.

Hu, W., Tran, V. N., Bulusu, N., Chou, C. T., Jha, S., and Taylor, A. 2005. The design and evaluation of a hybrid sensor network for cane-toad monitoring. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks (IPSN)*. 71.

Li, Q., Aslam, J., and Rus, D. 2001. Online power-aware routing in wireless ad-hoc networks. In *Proceedings of the ACM International Conference on Mobile Computing and Networking (MOBICOM)*.

Liu, J., Chu, M., Liu, J., Reich, J., and Zhao, F. 2003a. State-centric programming for sensor-actuator network systems. *IEEE Perv. Comput. 2,* 4, 50–62.

Liu, J., Liu, J., Reich, J., Cheung, P., and Zhao, F. 2003b. Distributed group management for track initiaition and maintenance in target localization applications. In *Proceedings of the 2nd International Workshop on Information Processing in Sensor Networks (ISPN)*.

Liu, T., Sadler, C. M., Zhang, P., and Martonosi, M. 2004. Implementing software on resource-constrained mobile sensors: experiences with impala and zebranet. In *Proceedings of the 2nd International Conference on Mobile Systems, Applications, and Services (MobiSys)*.

Luo, L., Abdelzaher, T., He, T., and Stankovic, J. 2006. Envirosuite: An environmentally immersive programming framework for sensor networks. In *ACM Trans. Emb. Comput. Syst.*

Luo, L., Cao, Q., Huang, C., Abdelzaher, T., Stankovic, J. A., and Ward, M. 2007a. Enviromic: Towards cooperative storage and retrieval in audio sensor networks. In *Proceedings of the 27th International Conference on Distributed Computing Systems (ICDCS)*. 34.

Luo, L., Huang, C., Abdelzaher, T., Stankovic, J. A., and Liu, X. 2007b. Envirostore: A cooperative storage system for disconnected operation in sensor networks. In *Proceedings of the Conference on Computer Communications (INFOCOM)*.

Lymberopoulos, D., Priyantha, N. B., and Zhao, F. 2007. mplatform: a reconfigurable architecture and efficient data sharing mechanism for modular sensor nodes. In *Proceedings of the 6th International Conference on Information Processing in Sensor Networks (IPSN)*.

Mainwaring, A., Culler, D., Polastre, J., Szewczyk, R., and Anderson, J. 2002. Wireless sensor networks for habitat monitoring. In *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*. 88–97.

Maroti, M., Kusy, B., Simon, G., and Ledeczi, A. 2004. The flooding time synchronization protocol. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys)*. 39–49.

Mathur, G., Desnoyers, P., Ganesan, D., and Shenoy, P. 2006a. Capsule: An energy-optimized object storage system for memory-constrained sensor devices. In *Proceedings of the 4th ACM Conference on Embedded Networked Sensor Systems (SenSys)*.

Mathur, G., Desnoyers, P., Ganesan, D., and Shenoy, P. 2006b. Ultra-low power data storage for sensor networks. In *Proceedings of the 5th International Conference on Information Processing in Sensor Networks (IPSN/SPOTS)*.

Puthenpurayil, S., Gu, R., and Bhattacharyya, S. S. 2007. Energy-aware data compression for wireless sensor networks. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*.

Sadler, C. M. and Martonosi, M. 2006. Data compression algorithms for energy-constrained devices in delay tolerant networks. In *Proceedings of the 4th ACM Conference on Embedded Networked Sensor Systems (SenSys)*.

Selavo, L., Wood, A., Cao, Q., Sookoor, T., Liu, H., Srinivasan, A., Wu, Y., Kang, W., Stankovic, J., Young, D., and Porter, J. 2007. Luster: wireless sensor network for environmental research. In *Proceedings of the 5th International Conference on Embedded Networked Sensor Systems (SenSys)*. 103–116.

SHAH, R. C., ROY, S., JAIN, S., AND BRUNETTE, W. 2003. Data mules: Modeling a three-tier architecture for sparse sensor networks. In *Proceedings of the IEEE Workshop on Sensor Network Protocols and Applications (SNPA)*.

SIMON, G., MAROTI, M., LEDECZI, A., BALOGH, G., KUSY, B., NADAS, A., PAP, G., SALLAI, J., AND FRAMPTON, K. 2004. Sensor network-based countersniper system. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys)*.

TILAK, S. AND ABU-GHAZALEH, N. B. 2005. Collaborative storage management in sensor networks. *Inter. J. Ad Hoc Ubiq. Comput. 1*, 1/2.

VASILESCU, I., KOTAY, K., RUS, D., CORKE, P., AND DUNBABIN, M. 2005. Data collection, storage, and retrieval with an underwater sensor network. In *Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems (SenSys)*.

WERNER-ALLEN, G., LORINCZ, K., JOHNSON, J., LEES, J., AND WELSH, M. 2006. Fidelity and yield in a volcano monitoring sensor network. In *Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*.

WOO, A. AND CULLER, D. E. 2001. A transmission control scheme for media access in sensor networks. In *Proceedings of the ACM International Conference on Mobile Computing and Networking (MOBICOM)*. 221–235.

XU, N., RANGWALA, S., CHINTALAPUDI, K. K., GANESAN, D., BROAD, A., GOVINDAN, R., AND ESTRIN, D. 2004. A wireless sensor network for structural monitoring. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys)*. 13–24.

ZEINALIPOUR-YAZTI, D., LIN, S., KALOGERAKI, V., GUNOPULOS, D., AND NAJJAR, W. A. 2005. Microhash: An efficient index structure for flash-based sensor devices. In *Proceedings of the 4th USENIX Conference on File and Storage Technologies (FAST)*.

Received May 2007; revised February 2008, June 2008; accepted June 2008