Audio-on-Demand over Wireless Sensor Networks

Hanhua Chen, Hai Jin, Lingchao Guo, Shaoliang Wu
Services Computing Technology and System Laboratory
Cluster and Grid Computing Laboratory
School of Computer Science and Technology
Huazhong University of Science and Technology
Wuhan, 430074 China
Email: {chenhanhua,hjin}@hust.edu.cn

Tao Gu
Department of Mathematics and Computer Science
University of Southern Denmark
Odense M, 5230 Denmark
Email: gu@imada.sdu.dk

***Abstract*—Audio represents one of the most appealing yet least exploited modalities in wireless sensor networks, due to the potentially extremely large data volumes and limited wireless capacity. Therefore, how to effectively collect audio sensing information remains a challenging problem. In this paper, we propose a new paradigm of audio information collection based on the concept of audio-on-demand. We consider a sink-free environment targeting for disaster management, where audio chunks are stored inside the network for retrieval.**

The difficulty is to guarantee a high search success rate without infrastructure support. To solve the problem, we design a novel replication algorithm that deploys an optimal number of $O(\sqrt{n})$ replicas across the sensor network. We prove the optimality of the energy consumption of the algorithm, and use real test-bed experiments and extensive simulations to evaluate the performance and efficiency of our design. The experimental results show that our design can provide satisfactory quality of audio-on-demand service with short startup latency and slight playback jitter. Extensive simulation results show that this design achieves a search success rate of 98% while reducing the search energy consumption by an order of magnitude compared with existing schemes.

***Index Terms*—audio-on-demand, wireless sensor network**

I. Introduction

The emerging wireless sensor networks (WSNs) have been revolutionizing the ways of collecting information from the physical world. The community has envisioned a large variety of applications, such as environment monitoring, scientific observation, and underground surveillance [4]. So far, audio represents one of the most appealing yet least exploited modalities in sensor networks, mainly because high-frequency audio sampling can produce extremely large data volumes over bandwidth-limited links.

In this paper, we investigate a new paradigm of audio services, namely audio-on-demand, in wireless sensor networks. We consider a sink/infrastructure free environment targeting for disaster management scenario. The target application is post-earthquake search and rescue, which becomes extremely significant after the hitting of recent constant violent earthquakes. When an earthquake occurs, recording and storing acoustic events and providing an on-demand retrieval service are essential to the later rescue in such systems based on the following observations: 1) The disaster area is often disconnected from the outside world, and 2) Most of the acoustic events are recorded before rescue could take place.

978-1-4673-1298-1/12/$31.00 ©2012 IEEE.

Since individual sensors are limited in their effective acoustic range, networks of acoustic sensors are needed to cover a disaster area. The requirements of a reliable audio-on-demand service are threefold. First, acoustic events should be recorded and stored inside the network because existing infrastructure if any may be destroyed in ruinous environmental conditions. Second, without a base station or other infrastructures, it is difficult if not impossible to efficiently locate acoustic events. Third, the non-disruptive on-demand playback of the audio anywhere in the network requires parallel data transferring and efficient buffer pre-fetching mechanism due to the limited bandwidth capacity of WSNs.

With the recent advances in NAND flash memory, new mote prototypes are now available that interface Mica-class processing and radio hardware to up to 8GB of flash memory [10]. The increasing in-network storage capability indeed makes the above store-and-fetch paradigm possible for WSNs, where the sensory data is stored inside the network and can be retrieved on-demand. However, existing infrastructure-free systems investigate only the "store" side of the store-and-fetch paradigm, leaving the other side an open issue. For example, EnviroMic [9] employs a distributed balanced storage mechanism to store the high-volume sensory acoustic event data inside the network. However, the sensory data stored inside EnviroMic cannot be readily accessed before all sensors are recovered from the experiment venue.

To support retrieval, a straightforward strategy is to locate the data using flooding. The problem is that flooding produces a large amount of traffic while the search success rate can not be guaranteed. It is natural to utilize replication strategy to improve search efficiency. However, how to achieve an optimal replication strategy with minimum retrieval energy consumption is not trivial for audio applications, especially under an infrastructure-free and bandwidth-limited wireless sensor network. To address this problem, in this paper we propose a novel replication strategy. We show that, by replicating both data replicas and query replicas uniformly at random across the network, the proposed strategy guarantees search success rate with a lower bound while the replication cost is $O(\sqrt{n})$, where n is the network size.

Based on the efficient replication strategy, we implement the buffer pre-fetching module in SAoD system, a real world audio-on-demand system over WSNs. SAoD uses time-division cooperative recording technique for collecting audio sensory data, where multiple sensors detecting the same acoustic event form a group with an elected leader to assign the time slots. Thus, the nodes in the group record the acoustic event cooperatively in turn. Consequently, different chunks of an acoustic event are naturally recorded and stored in the form of time addressable audio chunks by different nodes around the source of the acoustic event in different time slots.

Instead of replicating raw audio chunks, SAoD encodes the metadata of chunks residing on a node into a Bloom filter (BF). The metadata of each chunk includes the time addressable identifier. Thus, we compress the set of chunks into a space-efficient bit vector. By replicating the optimal number of the bit vectors across the network, SAoD further reduces the communication cost for the replication.

We implement the SAoD system with 30 IRIS motes equipped with MTS310 sensor boards. The experimental results show that SAoD provides high quality audio-on-demand service with very slight playback jitter and short startup latency. Results of extensive simulations in large scale networks show that SAoD's buffer pre-fetching can achieve guaranteed success rate while reducing the energy cost by one order of magnitude compared to existing schemes.

The main contributions of this work are threefold:

- We design and implement a real audio-on-demand system over WSNs and evaluate the performance using 30 nodes.
- We propose a novel replication strategy which guarantees chunk search success rate at minimized communication cost.
- We further reduce the communication cost of replication by encoding the chunk metadata using Bloom filter.

The rest of the paper is organized as follows. Section 2 reviews the related work. Section 3 introduces the probabilistic exhaustive replication model. Section 4 describes the detailed design of SAoD. Section 5 presents the system implementation and experimental results. Section 6 evaluates the design in large scale environment using extensive simulations. Section 7 concludes the paper with possible extensions.

II. RELATED WORK

Most of the existing work on audio sensor networks focuses on how to efficiently transfer the sensory data back to a base station (sink) [15] by either using online stream compression [13] or customizing high bandwidth sensor prototype [8].

In [15], Werner Allen et al. deployed 16 sensor nodes on the upper flanks of the Reventador active volcano, to collect the audio data. The nodes form a multi-hop routing topology and relay data via a long-distance radio modem to the observatory. They used the formed wireless sensor network to continuously sample acoustic data at the active volcano. A data collection protocol is designed to transfer continuously sampled acoustic data to the base station.

Wu et al. [13] tackled the problem of online compression of data streams in a resource-constrained network environment, where traditional compression techniques are not applicable. Particularly, they aimed at fast piecewise linear approximation methods with quality guarantee. They studied two versions of the problem which explore quality guarantees in different forms. For the error bounded piecewise linear approximation problem, they designed a fast online algorithms running in linear time complexity and requiring a constant space cost.

Xing et al. [8] designed and implemented a high bandwidth system for *quality-aware voice streaming* (QVS) in WSNs. QVS is built upon a new sensor hardware platform for high-rate audio communication. In their design they used the transceiver Chipcon CC1100 which has a 64bytes FIFO buffer and maximum data rate of 500 kbps. They used dynamic voice compression and duplication adaptation, and distributed stream admission control techniques. Their experimental results show that QVS delivers satisfactory voice streaming quality.

The above existing work on audio services over WSNs assumes the existence of a base station. The infrastructure-based schemes, however, may be problematic when applied to the audio-on-demand application addressed in this paper, because a user may hope to access only limited audio events of interest from any place in the WSN just as audio events are recorded everywhere. Transferring all the sensory audio data to a single base station is costly and infeasible. Moreover, a base station is a centralized point of failure. The failure of a base station in a disaster will paralyze the whole system.

To the best of our knowledge, we are the first to design and implement an audio-on-demand system over WSNs. The proposed retrieval scheme based on replication is different from existing flooding and GHTs [12]. Flooding does not guarantee the success rate without exhaustively search all the sensor nodes. The GHT partitions the name space over the nodes and has good search success rate for key-value search, while it suffers from the problem of exact match. Furthermore, although the problem of node failure for a key in GHT can be alleviated by using more than one node for a key, GHT can not survive the catastrophic failure. However, the case that a large number of nodes may be destroyed is norm rather than the exception in the target application in this work.

III. MODEL

The conventional wisdom for searching in a wireless sensor network without infrastructure support, such as flooding algorithm, is to replicate the query onto many nodes, which then evaluate the query on the data they store. While this works, it does not scale: to guarantee the search success rate, exhaustive search would require replicating the query onto every node. SAoD takes a different approach. It performs exhaustive search probabilistically. Before going to the details of the design of SAoD, we specify the model of chunk search in SAoD.

A. Replication Model for Search

Let the replicas of a data be red balls and the replicas of a related query green ones. Query matching in a WSN is

similar to a procedure of tossing the sets of green and red balls into a set of bins. Intuitively, with random casting, if the number of red balls or green balls is large enough, a collision of two kinds of balls in some bins can be guaranteed with high probability [6].

Theorem 1: Given a sensor network with n nodes, if we replicate r copies of a data and g copies of related queries, both uniformly at random in the network, the probability that at least one sensor node has both a data replica and a query replica is not less than $1 - \mathrm{e}^{-\frac{rg}{n}}$.

Proof: If all the replicas are distributed uniformly at random, for any given replica (data or query), the probability that it falls in a given sensor node is $\frac{1}{n}$, while the probability that it is not in the given sensor node is $1 - \frac{1}{n}$.

After all the number of r replicas are distributed uniformly at random, for a given sensor node, the probability that the node has no data replicas is $(1-\frac{1}{n})^r$. Thus, for a given query replica, if it is deployed in a sensor node, the probability that the node has no data replicas is $(1 - \frac{1}{n})^r$.

Because all the query replicas are distributed independently, the probability that none of the nodes with at least one query replica have any data replicas is

$$((1 - \frac{1}{n})^r)^g = (1 - \frac{1}{n})^{rg} \tag{1}$$

Thus, the probability that at least one node has both a data replica and a query replica can be computed by:

$$P = 1 - (1 - \frac{1}{n})^{rg} \tag{2}$$

For the given function $f(n) = (1 - \frac{1}{n})^{-n}$, we have

$$\begin{aligned} &\lim_{n\to\infty} f(n) = \lim_{n\to\infty} (1 - \frac{1}{n})^{-n} \\ &= \lim_{n\to\infty} [(1 + \frac{1}{n-1})^{n-1} \cdot \frac{n}{n-1}] \\ &= \lim_{n\to\infty} (1 + \frac{1}{n-1})^{n-1} \cdot \lim_{n\to\infty} \frac{n}{n-1} \\ &= \mathrm{e} \cdot \lim_{n\to\infty} \frac{n}{n-1} = \mathrm{e} \end{aligned} \tag{3}$$

It can be proven that $f(n)$ is a decreasing function on n. According to Eq.(3) we have $(1-\frac{1}{n})^{-n} \geq \mathrm{e}$. After performing some simple operations, we can have

$$\begin{aligned} &1 - \frac{1}{n} \leq \mathrm{e}^{-\frac{1}{n}} \\ &(1 - \frac{1}{n})^{rg} \leq \mathrm{e}^{-\frac{rg}{n}} \\ &1 - (1 - \frac{1}{n})^{rg} \geq 1 - \mathrm{e}^{-\frac{rg}{n}} \end{aligned}$$

Thus $P \geq 1 - \mathrm{e}^{-\frac{rg}{n}}$ is proved. ■

Theorem 1 shows that given a number of r data replicas and a number of g query replicas, if SAoD deploys all the replicas uniformly at random across the sensor network, the lower bound of the search success rate of a chunk is $1-\mathrm{e}^{-\frac{rg}{n}}$. Given a network size, the lower bound of search success rate is determined by the product of numbers of data replicas and query replicas.

It is clear that the model works when replicas are deployed uniformly at random across the sensor network. We will show how we sample random node and deploy replicas efficiently across the network in detail in Section 4.4.

B. Minimizing the Communication Cost

Given a fixed bound of success rate, it is important to save replicating cost by controlling the number of replicas to deploy. In Theorem 2 we show that the optimal number of replicas is determined by $\sqrt{n}$, where n is the network size.

Theorem 2: Given the size of the data replica is S_d *while the size of the query replica is* S_q*, the optimal numbers of data replicas and query replicas are in proportion to* $\sqrt{n}$*, where* n *is the size of the network.*

Proof: Let $\mathrm{e}^{-\frac{r \cdot g}{n}} = 1 - p = \mathrm{e}^{-c^2}$, we then have $r \cdot g = n \cdot c^2$, where $c = \sqrt{-\ln(1-p)}$.

Intuitively, the replicating cost is determined by the number of replicas. The optimal numbers of replicas should be achieved with the minimized cost.

$$\begin{cases} \min \ \ Cost = r \cdot S_d + g \cdot S_q \\ s.t \quad r \cdot g = n \cdot c^2 \end{cases} \tag{4}$$

The cost is minimized with the optimal values of r and g

$$\begin{cases} r = c\sqrt{n \cdot \frac{S_q}{S_d}} \\ g = c\sqrt{n \cdot \frac{S_d}{S_q}} \end{cases} \tag{5}$$

The result shows that the optimal numbers of data/query replicas are in proportion to $\sqrt{n}$. Thus proved. ■

IV. SYSTEM DESIGN

In this section, we present the design of SAoD. We first briefly describe how the audio events are recorded and stored. Then, we introduce how SAoD replicates the metadata of audio chunks in compressed form. Finally, we describe the replicating and chunk discovery scheme.

A. Cooperative Recording

SAoD utilizes the cooperative recording scheme proposed in [9] to split the task of recording an acoustic event into units divided by time slots among multiple sensors around the acoustic event. When multiple nodes detect the same acoustic event simultaneously, they form a group. The members of the group coordinate to elect a leader, who assigns recording tasks to individual members in turn. Thus, an acoustic event file, a_i, is naturally segmented on time. Audio is partitioned into chunks of uniform unit to make the file addressable on time. Each chunk has a fixed playing time equal to the length of a time slot. Due to the limited memory capacity of sensor nodes and the heavy loads caused by microphone sampling, choosing a proper size of the slot is not trivial. In Section 5, we will show how we obtain the optimal setting of time slots in SAoD system in greater detail. By using the cooperative recording technique, the chunks of an acoustic event file are naturally collected by different sensor nodes and stored in a distributed way. The time-division cooperative

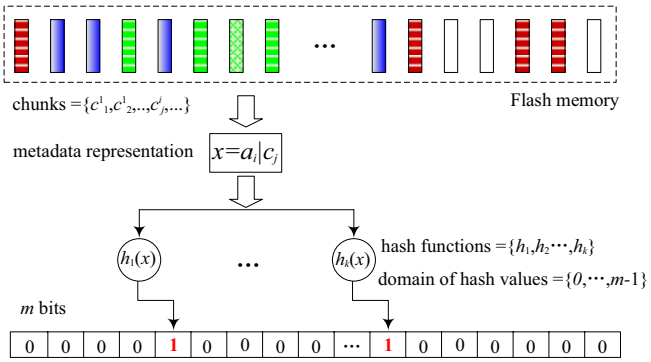

Fig. 1. Metadata encoding

recording scheme makes audio chunks time addressable as well as reduces the redundancy of chunk storage. It also effectively achieves a better load balance of storage as well as improves the data transferring performance during retrieval.

B. Metadata Encoding

Instead of replicating the raw audio chunks, we use Bloom filters [2] to encode the metadata of the chunks residing on a node. By replicating the metadata in a space-efficient way, SAoD greatly reduces the communication cost. In SAoD each sensor stores chunks of different acoustic events in the equipped flash memory. Figure 1 illustrates the process that a node in SAoD encodes the metadata of its chunks. On the top of the figure, the bars with the same mark denote the chunks of the same acoustic event file. For simplicity, we use c_j^i to denote the jth chunk (denoted by c_j) of the ith acoustic event file (denoted by a_i). SAoD obtains a global unique time addressable identifier of chunk c_j^i by combining the identifier of the acoustic event file and the sequence number (time address) of the chunk, namely $a_i|c_j$. By inserting all the identifiers of the chunks in the flash memory into the Bloom filter, a node in SAoD achieves a space-efficient bit vector for representing its chunks, which supports membership queries. The size of the Bloom filter can be determined by $m \geq \frac{kv}{\ln 2}$ [5], where k is the number of hash functions used in the Bloom filter and v is the maximum number of chunks limited by the capacity of the flash memory.

C. Network Size Estimation

As aforementioned, the optimal numbers of replicas are determined by the network size n. Without a base station, it is difficult to obtain such statistics [17]. To solve this problem, we can utilize a variant of the gossip algorithm first proposed in [11] to estimate the network size. The robust algorithm enables every node to quickly collect the global statistics in the network. The main idea of the method is as follows. Initially, each node does the following experiment: it flips a coin up to l times and counts the number of times the head appears before the first tail. It saves this count r in a bit vector (all bits initially set to 0) by setting the rth (counting from the right end) bit of the vector to 1. Then each node performs gossip. During each round of gossip, each node randomly selects a neighbor and sends its bit vector to the selected neighbor. The node receiving the bit vector performs a bitwise-or operation between the received bit vector and its local bit vector, and replaces the local bit vector with the resulting bit vector. The robust gossip scheme leads the computation of aggregated information to converge exponentially: after $O(\log n)$ rounds of gossip, all nodes will get the estimated network size with high probability.

D. Replica Deployment

As shown in Theorem 1, the optimal replication model requires that the replicas are deployed at random. In SAoD deployment, we assume that the sensors are deployed uniformly at random in a specified area. After SAoD computes the optimal number of replicas to deploy according to Theorem 2, it samples the optimal number of random locations in the fixed area and deploys the replicas of the metadata bit vector to the nodes nearest to the locations. Before casting all the replicas to the set of selected nodes, it forms a minimal spanning tree among the randomly sampled/computed nodes. Here, the logical neighbors in the minimal spanning tree communicate with each other using the underlying geographic routing algorithms [3]. The replicas are deployed using multicast along the minimal spanning tree. Figure 2 illustrates an example of the process of replica deployment across a rectangle with 4×10^4 sensors deployed uniformly at random. The bright green nodes are the ones with replicas deployed through the minimal spanning tree. Note, the location information of the source node is attached with the Bloom filter for chunk downloading during retrieval. Algorithm 1 describes the metadata replication strategy in detail.

E. Query Evaluation

During playback, the SAoD buffer pre-fetching module issues queries asking for the set of missing chunks in the pre-

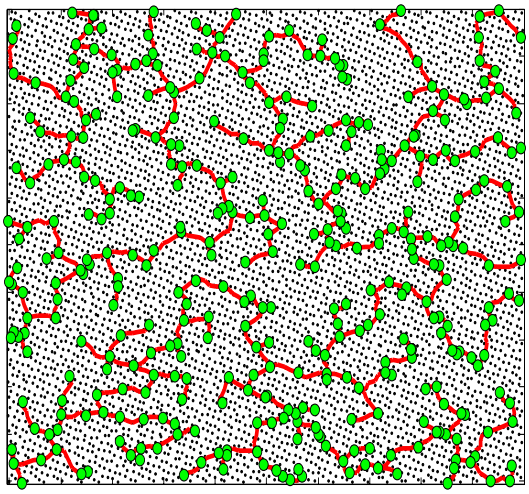

Fig. 2. Replica deployment

fetching window. SAoD replicates the query replicas to an optimal number of randomly and uniformly sampled nodes in the similar way shown in Algorithm 1. Every receiving node checks all the Bloom filters replicated locally. The member verification mechanism of the Bloom filter can effectively enable chunk discovery. If any chunk is tested to be contained in a Bloom filter, the replica is assumed to be the desired result with high probability. The $location_p$ attached with the chunk identifier is then returned for the buffer pre-fetching module to access the chunk. Note, due to the false positives of Bloom filters, the returned results may contain undesired ones with very low probability. This may lead to a slight decrease of the precision of the final results while keeping the recall rate guaranteed. If a false positive result is returned, SAoD filters it by sending the query to the node to evaluate the query locally. If no matching results, no actual chunks will be transferred. In Section 6, we will further discuss how to adjust the Bloom filter settings to achieve the tradeoff between the precision and the communication cost. It is not difficult to see that the chunk discovery algorithm can achieve better recall and latency than expected by Theorem 1, because the un-designated nodes on a path forwarding the queries can also provide data if they have. Algorithm 2 describes the query evaluation process in detail.

V. SYSTEM EXPERIMENTS

We have implemented SAoD system over a real testbed. The wireless sensor testbed consists of 30 IRIS motes all equipped with a MTS310 sensor board and running TinyOS. The IRIS mote is a widely available, low-cost and low-power platform. It basically inherits the MICAz mote with the improvement of a larger RAM (from 4KB to 8KB) and a longer range (from 100m to 300m outdoor). Each IRIS mote has a 512KB flash memory for storing audio data. Although the flash memory is limited, it is sufficient to evaluate the performance of SAoD.

Figure 3 illustrates the prototype system, deployed uniformly in a 5×6 grid with 30 nodes in the open field of a local 15m $\times$ 15m mini sports stadium. To set up the testbed, each node is equipped with a microphone capable of acquiring acoustic events. An audio source is broadcasted from a random location within the grid area. The sensors record the acoustic events using the cooperative recording technique in a time-division manner. All the chunks are stored into various nodes in the network. A node (i.e., IRIS mote) is placed randomly in the area to issue queries and gather acoustic chunks to feed the buffer of the player in the connected laptop. In the experiments, we let sensors start to work on the time of deployment. In the target application, the node of SAoD system can be triggered by the sensor's accelerometer module [16] because the vibration of the sensor is likely to be the indication of the happening of earthquake.

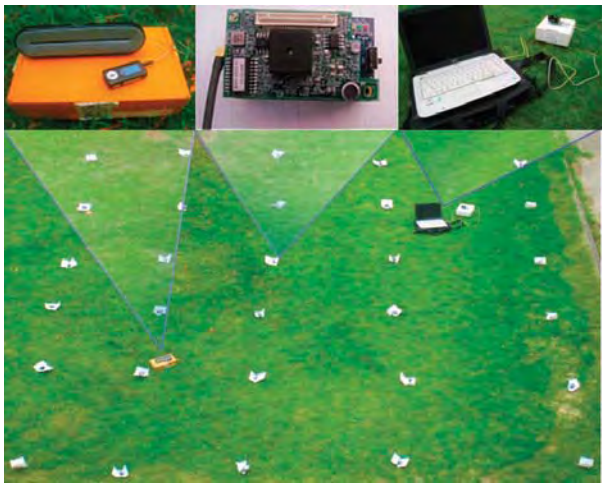

Fig. 3. System deployment

The key parameter in the system design is the chunk size, namely the length of time slot during cooperative recording. In the system implementation, we found it highly related to the memory size (8KB) of IRIS mote and the sampling frequency of the sensor board's microphone (20Hz to 16KHz from spec and frequency less than 5KHz is good as observed). During the experiment, we found when sampling frequency is over 7KHz, the failures surge in flash write. This is because sampling and flash write cannot perform concurrently. The flash write can not start before the sampling operation is over. To push the system performance examination close to the system limits, we set the sampling frequency to 7KHz by setting the MTS310 sensor board's parameter SAMPLE_INTERVAL to 143μs. Accordingly the time slot is set to 700ms, within which the program memory is almost full at 7800B.

Algorithm 1 Metadata Replication

Require: $EstimatedNetworkSize = n$ is achieved;
1: create an empty bit vector with m bits for node p, BF_p;
2: **for all** chunks in the local flash memory of node p **do**
3: insert the identifiers of the chunks into BF_p by using the set of hashing functions $\{h_j(\cdot), 1 \leq j \leq k\}$;
4: **end for**
5: compute r, the optimal number of replicas according to Theorem 2 using the gathered statistics n;
6: multicast BF_p attached with $location_p$, the location of node p, through the minimal spanning tree formed with the nodes nearest to the number of n_0 sampled locations;
7: **return**

Algorithm 2 Query Evaluation

1: $R \leftarrow \emptyset$;
2: **for all** BFs replicated in a node **do**
3: **for all** desired chunk t in the query Q **do**
4: **if** $\forall(j)(1 \leq j \leq k) s.t. BF_p[h_j(t)] = 1$ **then**
5: $R \leftarrow R \cup \{(location_p, t)\}$;
6: **end if**
7: **end for**
8: **end for**
9: **return** R.

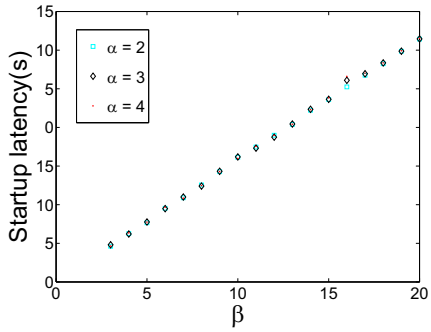

Fig. 4. Startup latency

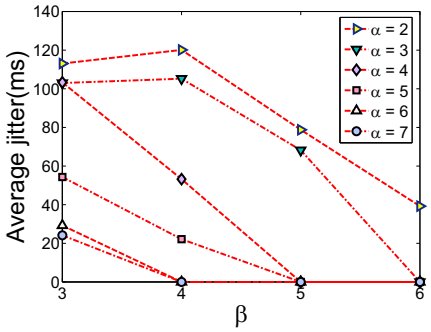

Fig. 5. Average jitter changes with β

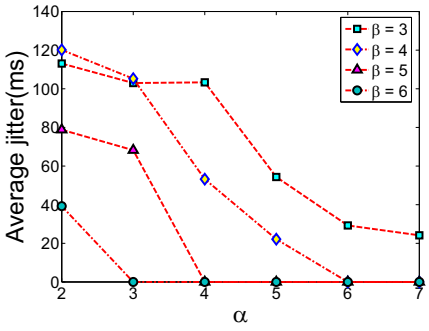

Fig. 6. Average jitter changes with α

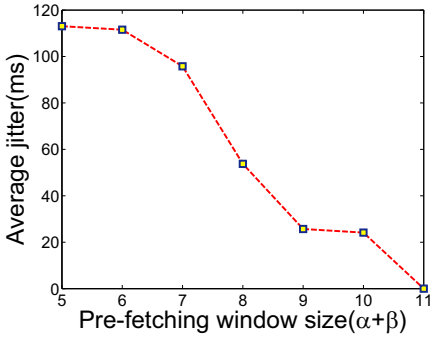

Fig. 7. Average jitter changes with α+β

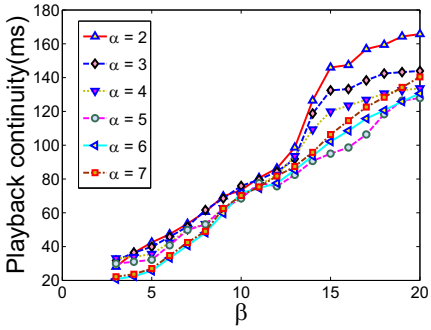

Fig. 8. Continuity changes with β

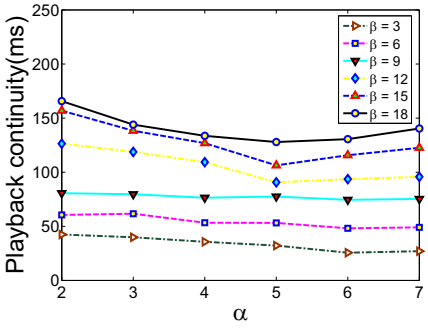

Fig. 9. Continuity changes with α

In the experiment, the chunks are pre-fetched from the network by using the query evaluation algorithm in Section 4 and fed to the buffer for the player. In the prototype, the client node performs the following scheduling algorithm. Every time slot, the scheduler checks the next β chunks. If all the chunks are available, it starts/continues playing while keeping fetching the chunks in the pre-fetching window with the size of $\alpha+\beta$ chunks. If not, it stops and fetches the missing chunks in the next $\alpha+\beta$ chunks. Once the next β chunks are all available, it resumes the playback. For smooth playback, the player pulls the fetched chunks from the head of buffer list at the playback speed. It is clear that the buffer settings influence the system performance and quality of service. We vary the parameters α and β in the scheduler to optimize the buffer settings.

In the experiments we mainly examine the quality of service of SAoD. We consider three metrics, startup latency, jitter, and playback continuity.

Startup latency quantifies the waiting time from the audio event is requested until the playback starts. In SAoD design, startup latency is an important metric because in the target applications such as earthquake rescue, a quick response from the system is critical. In the experiments, we adjust the buffer settings and check the startup latency.

Jitter quantifies the waiting time during playback. In the target earthquake rescue application, an acceptable playback quality with slight jitter is important for estimating the situation and determining the rescue plan for different cases. We can also use the waiting time per chunk to compute average jitter. As aforementioned in the above scheduling algorithm, if any chunk is missing in the next β-chunk window in the buffer, the playhead stops for buffer pre-fetching until the next $\alpha+\beta$ chunks are all available.

Playback continuity is defined as the total delay time divided by the number of played chunks. Here, delays include startup latency and all playback jitters. We ignore the cases requiring the seek and pause operations in this prototype.

$$Continuity = \frac{startup\ latency + playback\ jitters}{number\ of\ played\ chunks} \quad (6)$$

The unit is seconds per chunk played. Lower continuity values are better, representing better playback continuity.

Figure 4 shows that the startup latency increases nearly linearly when the parameter β increases, while the startup latency has very slight change with different settings of parameter α. Thus, we prefer to choose a smaller β for a short startup latency. However, Fig. 5 indicates when β is less than four, the average playback jitter is much worse and such worse jitter can be reduced by increasing α.

As we can see in Fig. 7 when the entire pre-fetching window size increases to 11, the average playback jitter decreases to zero. Figure 6 shows that the optimal settings of the buffer parameter is $\beta = 5$ and $\alpha = 4$, which can achieve the minimized buffer size with no playback jitter and short startup latency.

Figures 8 and 9 indicate that the playback continuity is dominated by the parameter β. This reveals that most of the user waiting time in SAoD system is startup latency. Due to

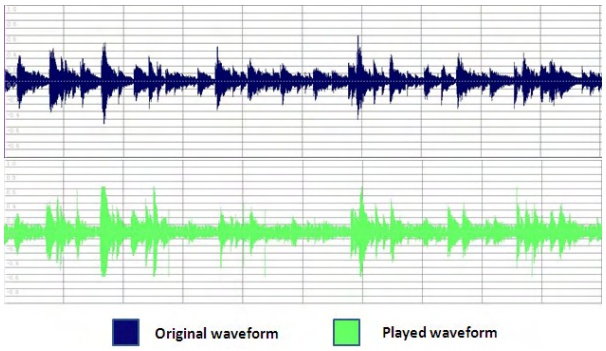

Fig. 10. Playback waveform

the slight jitter, startup latency is the most important factor and have much heavier weight in the continuity computation.

Figure 8 shows that with a higher setting of β ($\beta \geq 10$), the best choice of α is a moderate value α = 5. A smaller α, such as α = 2, incurs frequent short stops. On the other hand, a larger α, such as α = 7, leads to fewer but longer stops.

Based on the above analysis, in the SAoD system we set β = 5 and α = 4. The experimental results reveal that playback is quite fluent with very slight jitter and short latency.

Figure 10 shows the waveform of a traditional Chinese music named "Liangzhu" played on guitar, which is used in the experiment. We use guitar because its frequency range (70-1KHz) is in good agreement with that of human voice (75-1KHz), which we aim to record in the target application. Moreover, the audio frequency throughout a single guitar music varies in much wider range than that of a voice from a person. The conservative selection of guitar indeed pushes the system performance examination close to its limits. The higher part of Fig.10 is the waveform of the original music, while the lower part is what played by SAoD. Result shows the waveform of SAoD reproduces that of the original one, indicating high quality of the audio-on-demand service.

VI. SIMULATIONS

In this section, we evaluate the design using simulations in large scale networks. We first introduce our simulation methodology and the setups. Then we specify the metrics used for the evaluation. At last we present the results comparing with existing schemes. In the comparison we use the flooding scheme as the baseline, since it is extensively used in the literatures. To make a more fair comparison, in the baseline scheme we also replicate the data in the same way as SAoD.

A. Simulation Setups

In the simulation, we put $10,000$ nodes on the grid of a 1,000m $\times$ 1,000m rectangle and then perturb each point by a random shift following a normal distribution with $\sigma = 3$, which has been widely treated as an approximation for the manual deployment of sensor nodes [4]. We use the unit disk graph model [7] for sensor communications. Although the unit disk model does not perfectly simulate the empirical observations in practice, it can be considered as a clean abstraction ignoring any uncertain physical disruptions and is sufficient for validating the principle of this design. Indeed, it can be considered as a conservative measure. For example, we can set a threshold to get a conservative communication range by taking the lower bound in some non-disk communication cases.

The communication radius of each node is set to 50m. We randomly deploy a number of $10\sqrt{n}$ audio sources. Each source can be detected within a distance following a normal distribution with $\mu = 19$ and $\sigma = 2$. The length (ms) of an audio event follows a normal distribution with $\mu = 2,000$ and $\sigma = 500$. We simulate the cooperative recording by randomly assigning the audio chunk among the nodes within the radius the audio source can be sensed. According to the de facto standard used by TinyOS [1], every message in our design has a limited length of 46 bytes with 28 bytes payload, 11 bytes header information and 7 bytes metadata. During the search process, we randomly select a node to issue queries.

B. Metrics

We evaluate the performance of SAoD using the metrics described as follows.

Energy consumption: Communication tends to dominate the energy consumption on sensor node in conventional wisdom. Saving energy is critical in SAoD design. We compute the energy consumption for transmitting data M_i by: $E_i = \frac{|M_i|}{L} H_i$, where $|M_i|$ is the size of the transmitted data, L is the length of the payload in each packet and H_i is the hops the data transmitted from the source to the destination. The practical energy cost depends on the underlying sensor platform. The transceivers of current IRIS mote have a data transfer rate of 250 Kbps. A mote consumes 29mW power for receiving and 42mW power for transmitting (at 0dBm) [14].

Success rate: In this design, the chunk retrieval success rate is critical to the quality of audio services. Success rate is defined as the fraction of queries which have returned matched chunks if exist. Ensuring a high success rate is the main challenge in the design of a sink-free audio-on-demand WSN.

Energy efficiency: We define the energy efficiency as the ratio of energy consumption to the success rate. Compared with the energy consumption and the success rate, the energy efficiency is a more fair metric. Lower energy efficiencies represent greater success rate with less energy consumed.

Search latency: A short search latency is always desirable in the audio-on-demand service in WSNs. It is defined as the duration between the time when a query is issued and the result is returned. It is the sum of the latency during each hop in the underlying wireless links. The time required to transfer a packet includes the propagation time as determined by the distance between the underlying nodes and the transmission time. The overall transmission time is computed by: $\frac{M}{B} + \frac{L}{s}$, where M is the size of the transmitted data (bits), B is the bandwidth (bps) of a wireless link, L is the distance between two communicating nodes, and s is the propagation speed in medium. In the SAoD design, the playhead moves ahead

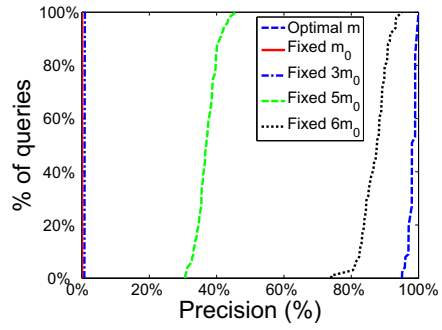

Fig. 11. Optimal settings of m in Bloom filters

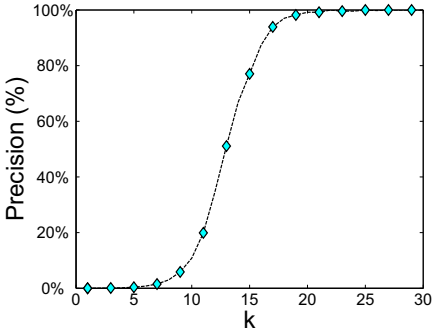

Fig. 12. Optimal setting of k in Bloom filters

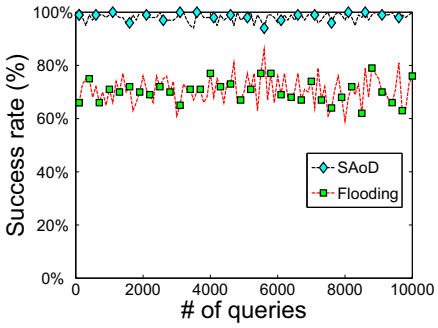

Fig. 13. Success rate

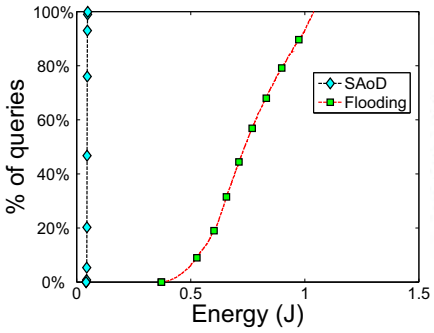

Fig. 14. Energy consumption

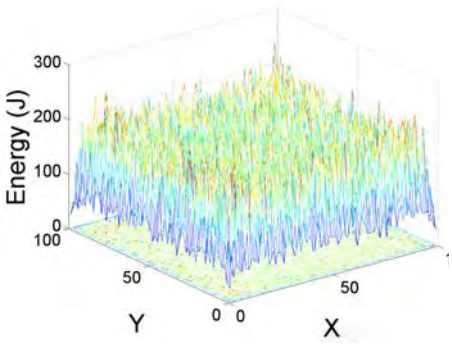

Fig. 15. Load balance

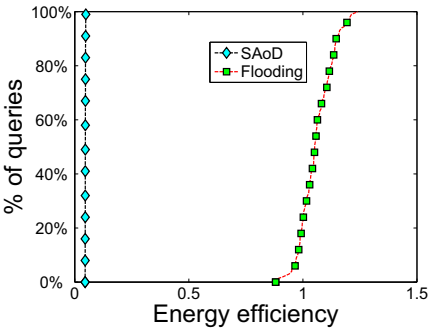

Fig. 16. Energy efficiency

only when the buffer is full. Fast locating and pre-fetching the chunks are quite important to the system.

Precision: Precision captures the fraction of relevant chunks in the returned results. Due to the false positives of a Bloom filter, some returned results may not be the desired results. If a false positive result is returned, SAoD needs to evaluate the query by sending the query to the node holding the raw data.

C. Simulation results

In the baseline flooding scheme we set the value of *Time to Live* (TTL) to 25. When simulating SAoD scheme, we set $\lambda = 4$ .To make the comparison fair, the flooding is performed with the same data replica deployment as that of SAoD.

It is clear that the settings of the Bloom filter parameters including m and k are critical to the performance of our scheme. We first vary the setting of k from 1 to 30 and find that when $k = 19$ the false positive of a Bloom filter is quite acceptable. Figure 12 plots how the precision changes with the parameter k in detail.

Since different nodes in the network may have different numbers of chunks, the m of the Bloom filter should vary according to this number. We set the value of m by: $m = \frac{vk}{lg2}$, which achieves the slightest false positive of Bloom filters, and compare the precision with the case with fixed m_0 setting throughout the network, where m_0 is the average value of the different m values chosen above. This means that using fixed m_0 throughout the network and using optimized m values in different nodes will have the same replication energy cost. Results in Fig. 11 show that with the optimal m value in each nodes, we can achieve an average precision of 98.35% while the precision with a uniformed m_0 is only 0.016%.

From the above results, we set k to 19 and compute an optimal m in the following experiments.

Figure 13 shows the search success rate. It indicates that the SAoD scheme greatly outperforms the baseline scheme. The average search success rate of flooding is 70.8% while the search success rate of SAoD is 98.1%. The results show that the SAoD achieves perfect chunk discovery success rate, which is critical for our target application. In practice, we can adjust the parameter $\lambda = \frac{rg}{n}$ to tradeoff between search success rate and resource consumption.

Figure 14 examines the energy consumption of SAoD. The lowest energy consumption of the flooding scheme is about 0.37J while the highest energy consumption of SAoD is 0.05J. The average energy consumption of a query using flooding is 0.75J, while the average energy consumption of a query using SAoD is 0.046J. The results show that the SAoD replication strategy is indeed cost-efficient. It reduces the energy consumption to more than one order of magnitude lower than that of flooding scheme.

Figure 15 shows the loads of all the nodes in the network. It shows that the loads are balanced among the nodes in

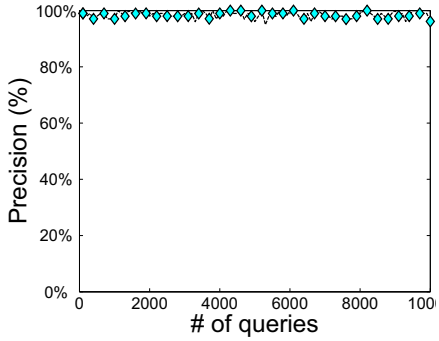

Fig. 17. Precision

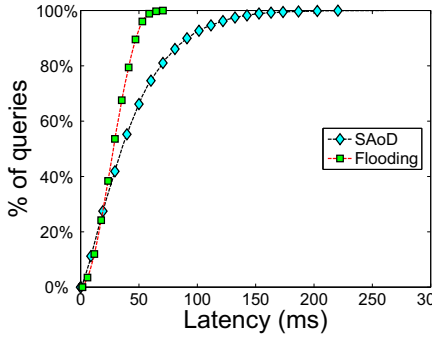

Fig. 18. Latency

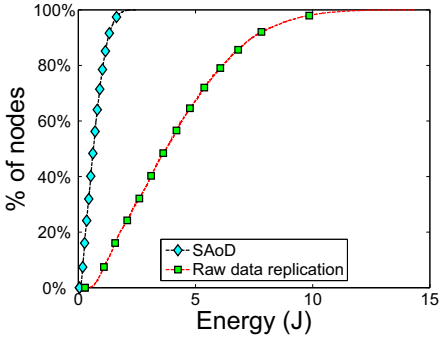

Fig. 19. Energy efficiency

the SAoD network. This is important for a wireless sensor network, because the overload of some sensor nodes will make them die quickly and this can paralyze the entire network.

Figure 16 shows that the average energy efficiency of SAoD query is 0.0466 while that of flooding is 1.0594. This means a 23× performance improvement.

Figure 17 illustrates the precision of SAoD. It shows that average precision of SAoD query is 98.4% which is quite acceptable. Although due to the false positives of Bloom filters, the system can not achieve a precision of 100%, the cost of removing the false positive is cheap (Section 4.5).

Figure 18 plots the latency of SAoD. It shows that 27.6% SAoD queries have shorter search latency than those of flooding, while overall the latency of SAoD is slightly larger. This is because SAoD utilizes the minimal spanning tree to multicast the query replicas while the flooding scheme follows the shortest path to propagate the queries. The average latency of SAoD is 44.6 ms while the average latency of flooding is 29.4 ms. The results show that in the 1000m × 1000m area with 10000 nodes the longest latency of SAoD is about 262ms. Such a low latency is quite acceptable in real-world systems.

Figure 19 shows the data replication costs. We assume that each chunk identifier takes 20 bytes. It shows that by replicating Bloom filters, the SAoD scheme greatly reduces the energy cost for replicating.

VII. CONCLUSIONS AND FUTURE WORK

In this paper, we design and evaluate SAoD, an audio-on-demand system over WSNs. SAoD stores the audio data inside the network for retrieval. We show mathematically that SAoD achieves perfect success rate for chunk retrieval at the cost of $O(\sqrt{n})$. Instead of replicating the raw audio data, we use Bloom filters to compress the metadata of chunks. We implement a real system based on IRIS mote as well as conduct comprehensive simulation to evaluate this design.

Equipped with the results obtained in this study, we see great potentials of applying SAoD to a wide range of audio-on-demand applications. Meanwhile, there are also some interesting open questions for our future work. For example, the potential breakdown of motes may lead to loss of chunks. We will investigate distributed chunk-grain data redundancy and recovery mechanism, in our design in the future.

ACKNOWLEDGMENT

This work was supported by the National Science Foundation of China (NSFC) under grant 61003006 and a grant from the Ph.D. Programs Foundation of Ministry of Education of China (No.20110142120080) .

REFERENCES

[1] "Tinyos: http://www.tinyos.net/," 2008.
[2] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communication of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.
[3] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia, "Routing with guaranteed delivery in ad hoc wireless networks," in *DIAL-M*, 1999.
[4] H. Chen, M. Li, H. Jin, Y. Liu, and L. M. Ni, "Mds: Efficient multi-dimensional query processing in data-centric wsns," in *RTSS*, 2008.
[5] L. Fan, P. Cao, J. M. Almeida, and A. Z. Broder, "Summary cache: a scalable wide-area web cache sharing protocol," *IEEE/ACM Transactions on Networking (TON)*, vol. 8, no. 3, pp. 281–293, 2000.
[6] R. A. Ferreira, M. K. Ramanathan, A. Awan, A. Grama, and S. Jagannathan, "Search with probabilistic guarantees in unstructured peer-to-peer networks," in *P2P*, 2005, pp. 165–172.
[7] S. Funke, A. Kesselman, U. Meyer, and M. Segal, "A simple improved distributed algorithm for minimum cds in unit disk graphs," *ACM Transactions on Sensor Networks (TOSN)*, vol. 2, no. 3, pp. 444–453, 2006.
[8] L. Li, G. Xing, L. Sun, and Y. Liu, "Qvs: Quality-aware voice streaming for wireless sensor networks," in *ICDCS*, 2009.
[9] L. Luo, Q. Cao, C. Huang, T. Abdelzaher, J. A. Stankovic, and M. Ward, "Enviromic: Towards cooperative storage and retrieval in audio sensor networks," in *ICDCS*, 2007.
[10] K. Mihic, A. Mani, M. Rajashekhar, and P. Levis, "Mstore: Enabling storage-centric sensornet research," in *IPSN*, 2007.
[11] S. Nath, P. B. Gibbons, S. Seshan, and Z. R. Anderson, "Synopsis diffusion for robust aggregation in sensor networks," in *SenSys*, 2004.
[12] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker, "Ght: A geographic hash table for data centric storage," in *WSNA*, 2002.
[13] E. Soroush, K. Wu, and J. Pei, "Fast and quality-guaranteed data streaming in resource-constrained sensor networks," in *MobiHoc*, 2008.
[14] M. Srivastava, "Sensor node platforms & energy issues - a tutorial," in *MobiCom*, 2002.
[15] G. WernerAllen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh, "Fidelity and yield in a volcano monitoring sensor network," in *OSDI*, 2006.
[16] N. Xu, S. Rangwala, K. K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin, "A wireless sensor network for structural monitoring," in *SenSys*, 2004.
[17] B. Zhang, X. Cheng, N. Zhang, Y. Cui, Y. Li, and Q. Liang, "Sparse target counting and localization in sensor networks based on compressive sensing," in *INFOCOM*, 2011.