SensorEar: A Sensor Network Based Eavesdropping System

Ge Ruan, Soumya Jain and Sencun Zhu
The Pennsylvania State University
Emails: ruan@cse.psu.edu, ssj125@psu.edu, szhu@cse.psu.edu

Abstract— **We present a sensor network based eavesdropping system, *SensorEar*, which demonstrates a serious threat to an individual's privacy. The small size, long battery lifetimes and easy deployment enable the use of sensor networks in malicious ways. Our system comprises motes which record speech, compress the data and transmit it over a multi-hop network in real time. After noise reduction, we are able to hear, with sufficient clarity, the person's conversation. In contrast with the 10 - 100 Hertz sampling of typical applications, audio recording requires a sampling frequency of several kilohertz. We address research challenges associated with this high data volume system, and design solutions including a multi-rate compression algorithm, communication protocol modifications and a unique FDMA/TDMA based system architecture. We have built a complete prototype using commercial off-the-shelf components. Further, we have chosen the most basic hardware to make our solutions truly platform independent.**

I. Introduction

Improvements in wireless communication and processor technology have enabled the development of low cost, low power, small sized motes which have sensing, computation and radio communication capabilities.When deployed as a network, they can be used in a wide range of civilian and military applications. Typical deployments are in remote and inaccessible places to collect data regarding various natural phenomena, species of animals or environmental conditions. Sensor networks are also used in industrial and other forms of monitoring applications.

As an illustration of defense applications, consider a body wearable sensor network [16], which estimates the location of a snipper based on information recorded from the gunshot. In ecology research, scientists have deployed a sensor network to identify, track and measure the population of rare birds and their habitat conditions [10]. Similarly, a sensor network has been used for industrial monitoring [2]. Recently, researchers have started exploiting the use of sensor networks in our daily lives. For example, the CodeBlue [15] system outfits patients with motes, enabling nurses and doctors to continuously monitor their vital conditions and status remotely. Similarly, we could use a system to measure and track the performance of athletes [5]. All of the above mentioned applications show that sensor networks will enable many previously impossible research projects and bring more convenience to our lives, at the same time becoming increasingly pervasive.

The proximity and pervasiveness of sensor networks, however, raises the very important problem of privacy. In contrast with the traditional concept of privacy in computer science, this is a far more direct threat to personal privacy. Imagine the following scenario. A dishonest student programs a mote as a recording de- vice (using the microphone sensor) and places it in his advisors office. Then he drops more motes, programmed as routers on the path from the recording mote to his own office. This sensor network enables him to eavesdrop on all conversations in his advisors office! Another even more dangerous scenario is where a sensor network deployed for a valid application, could be used simultaneously for malicious purposes.

Three main features empower the use of sensor networks in such malicious ways. Firstly, they are "tiny" and thus difficult to detect. For example, the MICA2 mote from Crossbow Inc., which we use in our project, is just 5 x 3.2 x 0.7 (cm), weighing only 18 grams (excluding the battery pack). The Smart Dust project [13], conceptualizes motes to be grain-sized eventually. Secondly, the typical battery life of a mote is above one year [3], and thus no "maintenance" is required for a long time. Thirdly, sensor networks are commercially available as ready-to-be-deployed kits. The technology currently being in a development phase, they are somewhat expensive, but in the future are expected to cost less than $ 1 per mote [13].

In this paper, we prove the seriousness and feasibility of this "eavesdropping" threat. We present a complete working system with commercial off-the-shelf products. Further, we use only the cheapest and lowest capability platforms in the market. Other contributions are addressing the research challenges due to the contrast between the high data volume of audio applications, and relatively very low platform bandwidth. We present solutions involving radio stack modification, a novel multi-rate compression algorithm and a FDMA/TDMA based multi-hop system architecture for this application.

The rest of this paper is organized as follows. In Section 2, we outline the related work. We describe the design goals in Section 3. In Section 4 we explain the challenges faced due to the platform's limitations. Section 5 details the solutions developed, while the implementation is presented in Section 6. Finally, in Section 7 we analyze the performance from various perspectives, and in Section 8 discuss the system limitations and future work.

II. Related Work

To the best of our knowledge, there is no personal privacy relevant research in the area of sensor networks. Han et al.'s research in [7] is somewhat close. They detect human presence by sensing humidity of the environment. However information about only presence /absence is brief and coarse, and may not be crucial enough to pose a serious threat.

In 2007, Luo et al. [9] developed a distributed acoustic monitoring system called EnviroMic. Their application is somewhat similar, but the primary objectives and design goals are very different. Their project is aimed at exploring distributed storage in sensor networks, where the motes are equipped with a 512 MB flash memory to store all the sampled data, to be retrieved later. Our goals are to achieve real time performance, in which the recorded data is transmitted back as it is being recorded. Also, they utilize the MICAZ [9] platform, which has a maximum transmission rate of 250 kbps, while we use the MICA 2, with a maximum data rate of 19.2 kbps. These additional restrictions prompted us to develop compression algorithms and make our system extremely efficient by design itself. Further, if our prototype works well on MICA2 motes, then it will work better on more powerful platforms.

In [11], the authors described a real-time voice stream-capability in wireless sensor networks and summarized their deployment experiences of voice streaming across a large sensor network of FireFly nodes in an operational coal mine. FireFly has several integrated layers including specialized low-cost hardware, a sensor network operating system, a real-time link layer and network scheduling. Although achieving the same technical goal, our work differs from the above one is that we use the off-the-shelf commercial MICA2 motes instead of their own FireFly nodes. The radio of FireFly nodes runs 802.15.4, which has the maximum raw data rate of 250 kbps, while the maximum data rate of MICA2 is only 19.2 kbps. Thus, we are faced with the greater challenge for timely delivery of voice signals.

III. Design Goals

Clear Intelligible Sound The human voice lies in the frequency range of 300Hz to 3200Hz. According to Nyquist's Theorem, to achieve a perfect reproduction of the original sound, from samples, the sampling rate should be at least twice that of the highest frequency component in the signal. Telephony applications generally use an 8 kHz sampling frequency and that is what we aim to achieve.

Real-time System The eavesdropper / computer listening to the conversation should be able to hear the speech as it is being spoken, with only a minimal delay for transmission and processing. This implies that all the sound samples should be "removed" from the mote system in real time, as opposed to storing them temporarily (may be in an external flash) and retrieved later. Assuming an 8 kHz sampling frequency, and an 8 bits/sample ADC resolution, data is generated at 64 kbps, and must be transmitted away at the same rate. Further, any processing, both at the mote as well as at the base station needs to be fast enough to meet the real time performance requirements.

Multihop Capability We cannot realistically expect the eavesdropper to be within a one-hop range of the recording motes. The system should have a multi-hop functionality, thus, being able to transmit over fairly long distances. This implies designing an efficient communication protocol for the sampling motes, the intermediate routers (relaying motes) and the base station.

IV. Limitation of the Platform

We have chosen the MICA2 motes as our operating platform. They are the cheapest, most basic motes available in the market, with the lowest capabilities in terms of storage, processing and transmission. This implies that the research challenges we address here are completely platform independent, and will only work better on later generation motes. Further, while later generation motes today are more powerful, it is reasonable to expect that the first motes to become truly 'dust-sized' will be with low capabilities.

Towards achieving our design goals, the limitations of MICA 2 pose a lot of non-trivial challenges. Compared to the MICAZ, TelosB and other later generation motes, MICA 2 is much weaker in several aspects. The limitations of the hardware and software platforms, which motivate our project to be more innovative, are detailed below.

Low Level Microphone We use the MTS310 sensor board, which has a microphone. However, it is designed only to detect the presence / absence of sound, for example in an application like [16]. Correspondingly, the default TinyOS microphone-control module returns only a Boolean variable. Practically, this implies a lot of distortion and noise in the recorded speech, while at the same time does not allow us to go below a certain sampling rate.

Severe Memory Constraints As mentioned before, the recording motes will generate data at 64 kbps. MICA 2 has only a 512 KB external flash, which is however fairly slow in writing - it potentially has a 3 ms stabilizing time after each write. This implies firstly, that we cannot store the data temporarily for very long, and secondly any processing that has to be done on the data, needs to be with algorithms not requiring a large block of data to operate on.

Low Transmission Rate The maximum data rate provided by the MICA2 radio (CC1000 radio module) is 19.2 kbps (38.4 kbaud, with Manchester coding). Our ideal data rate required is 64 kbps. Overcoming this gap is the major challenge of the project.

V. Solutions

In this section we present the building blocks of our system. These techniques enable us to achieve the design goals, and overcome the limitations of the system.

A. Data Compression

The first solution we develop is the compression of the large volume of data generated. As mentioned in the design goals (Section 3.2), this rate is 64 kbps. The maximum data rate achievable by the MICA 2 radio is only 19.2 kbps. Further, with the default communication protocol stack, we are able to achieve only about 4 -5 kbps typically. Towards bridging this gap, the first building block of our system is a multi-rate speech compression algorithm.

There are four main objectives which our algorithm here must meet. Firstly, it must be simple enough to execute with the mote's limited resources. Secondly, it must run in linear time (O(n)) to satisfy the real time constraint of the entire system. Thirdly, its compression latency must be very low, i.e., it should not require the collection of a large block of samples before it can execute. This is important because of the mote's low memory resources, and the real time performance goals of the entire system. Finally, it must provide a high enough compression ratio to be able to bridge the gap between the high data generation rate and low transmission rate.

All generally used compression algorithms make tradeoffs between the above mentioned objectives. WinZip achieves a fairly high compression rate, but has a compression latency of 32 KB - using block sizes below this drastically reduces the compression rate. Typical audio compression algorithms like MP3 and AC-3 use the Modified Discrete Cosine Transform (MDCT) [14], which would be very difficult to run on the mote's processor, and do not have linear run times. In cell phone communication, AMR(Adaptive Multi-Rate Compression) [20] is widely used in GSM and UMTS. It uses link adaptation to select from one of eight different bit rates based on link conditions. Our study found that the complexity of the AMR algorithm is at least 5 times of our MRC algorithm.

Among the low bit rate speech encoding, the LPC(Linear Predictive Coding) [21] is one of the most common ones. It can yield a data stream with very low bit rate (2.4kbps), but its compression speed is too slow for our application. According to [21], the codec of LPC uses a bit rate of 2.4 kbit/s, requires 20 MIPS of processing power, 2 kilobytes of RAM and features a frame size of 22.5 ms. Additionally, the codec requires a large lookahead time of 90 ms. But according to [4] and [1], the microcontroller of Mica2 provides 1 MIPS per MHz while it runs at 7.37 MHz. In other words, the processing power of Mica2's CPU is 7.37 MIPS, which is not enough for LPC.

Thus, instead of using any existing algorithms, we chose to design our own compression algorithm, which would be specific to the type of data we are dealing with and make a balance between the compression rate and compression speed according to the constraints imposed by the platform.

Algorithm Details Broadly speaking, we use a combination of different algorithms (RLE [17] and PCM [12]), each yielding a different compression rate - thus the name Multi-Rate Compression (MRC).

Theoretically, the sampled data can range from 0 to 255. However, we observe that the majority of them lie between values of 116 to 122, with 119 corresponding to normal silence. Firstly, we subtract 119 from the samples. Secondly, we differentiate the data by taking the difference between successive samples. We now observe that the majority of values are between -3 and 3. Figure 1 shows the distribution for a typical data set. This distribution led us to designing three different techniques for each 'mode' of the data, i.e. one for 0s, one for absolute values less than value 1, and the other for absolute values less than 3. The percentage of absolute values above 3 is small enough to be ignored. We clip them into the third group by converting them to 3 or -3. For 0s, we apply RLE, i.e. counting the number of consecutive zeros and transmitting that value with an identifier. In this case, at most, we can compress 63 consecutive zeros into 1 byte; or non-zero values absolute under 1, we apply PCM for these values, with a 3 level quantification (0,1,2) and compress 4 samples into one byte; For non-zero values absolute values under 3, we apply PCM again, with a 7-level quantification (0 through 6) and compress 2 samples into one byte.

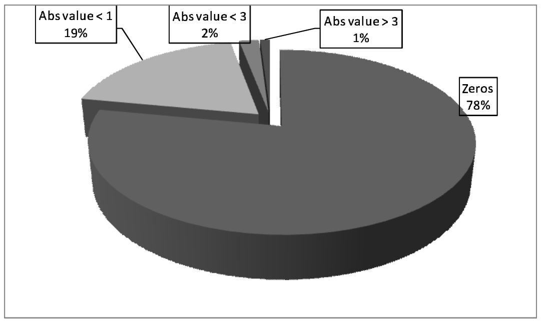

Fig. 1. Sample Distribution of Data

The data output from different compression schemes are mapped into different ranges before transmitting, and this is used at the receiver to identify which scheme was used, and run the corresponding decompression algorithm. The compression rates thus are between a minimum of 2 and a maximum of 63, and typically we achieve an average rate of 2.5 to 4.

TABLE I
AN EXAMPLE OF MRC

Input	0000	101	31
Status	$S0$	$S1$	$S2$
Intermediate Step	–	$(3-ary)$210	$(7-ary)$62
Result(Binary)	4(0000 0100)	21(0001 0101)	44(0010 1101)
Mapping(+)	192(1100 0000)	0 (0000 0000)	128(1000 0000)
Output	196(1100 0100)	21(0001 0101)	172(1010 1100)

Table 1 illustrates how MRC works on an input sequence 0,0,0,0,1,0,1,3,1. The input size is 9 bytes, while the output size is 3 bytes. So the compression rate is 3 here.

This algorithm satisfies all the requirements listed before.

- It is simple and requires no complex functions or processing.
- It runs in linear time, i.e. O(n).
- Its compression latency is extremely low - typically 1 byte.
- We achieve a compression rate of 2.5 to 3, which is enough for our application.

B. Optimization of Radio Protocol Stack

Transmission of the data back to the base station in real time is the key part of this project and perhaps the most

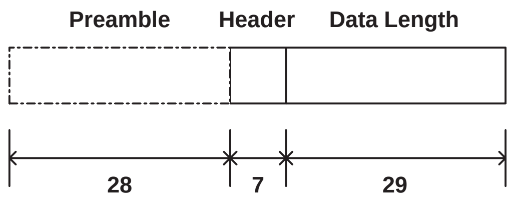

Fig. 2. Default Packet Length

challenging one, because the throughput is the bottleneck of the whole system. The data sheet of MICA 2 specifies the maximum transmission rate of the CC1000 radio to be 19.2 kbps. However, in practice we observe that this rate is never achieved with the default communication protocols. We could barely get a 4 - 5 kbps data rate in actual experiments. Thus, our second enabling technique is the optimization of the communication protocols, to achieve as close of the theoretical 19.2 kbps limit.

We find that the disparity in actual and theoretical data rates is primarily due to two reasons - overheads and medium contention. We address both these issues as described below.

Overhead Reduction: The default TinyOS packet length, as shown in Figure 2, is 36 bytes - 7 header bytes and 29 data bytes. Further, for synchronization a preamble of length 28 bytes is used before every packet. This implies that the effective payload is only 29 bytes out of 64 bytes per packet, implying a relatively low efficiency of 45%. To address this problem, we increased the payload size however to 247 bytes resulting in an efficiency of 87%, which is much higher. This achieves a 4 -5 kbps improvement in data rate.

Medium Contention: In TinyOS, CSMA/CA is applied as the MAC layer protocol. On sensing and finding the channel busy, the sender waits a random back off time, thus ensuring a low probability of collision. This timer is set to the sum of the packet length (247 in our application) and a random number picked from a uniform distribution between 0 and 127. Compared to the packet size, this waiting time is very high. Now, in our round robin scheduled system (detailed later in system architecture), only one mote transmits at a time implying that we do not need a very conservative MAC layer collision avoidance protocol. Thus, we reduce the range of the back off timer, by modifying the radio stack defined in the CC1000RadioIntM module. We can now achieve a data rate of about 15 kbps.

This scheme could raise concerns regarding higher chances of packet collisions. However, we experimented exhaustively to find that the percentage of packets lost is within 1 %. This agrees with our previous analysis - that since we have a round robin scheduling, we are already implementing a medium sharing mechanism, and the default conservative CSMA/CA is “overkill”.

C. *Collaborative Relaying and Channel Switching*

To realize eavesdropping from a long distance, we use a multi-hop system architecture. Intermediate motes function as relaying motes forwarding commands from the base-station to the sampling motes, and the recorded samples back from the motes to the base-station. The design of these intermediate relaying layers is an important part of the entire architecture. Firstly, we note that our effective data rate is at least 15 kbps, i.e., the sampling layer motes, after their sampling quantum, send out the compressed data at 15 kbps to the first relaying layer. The receiving relaying layer mote, now, can only receive the data, requiring the same amount of time to send it out to the next relaying layer. This is simply because we are using the maximum communication throughput (send and receive) available. This analysis shows that we need to effectively make the data rate twice that of the existing rate, at the relaying layers. There are two solutions intuitively. One is to store the data for a while, and send it later. This violates our real time constraints, as well as is not a good solution given that the external flash is only 512 KB, and thus will fill up quickly. The other solution is that we use multiple motes forming multiple paths in the relaying layers. Our prototype uses three motes in each relaying layer. Each mote listens to and transmits every third packet, based on the sequence numbers of the packets. This system is extended similarly to the other routing layers as well. To avoid collisions between packets sent between the various layers, we employ FDMA between layers, i.e., each inter-layer communication uses a different channel. Thus, each mote listens for packets on a channel from its preceding layer, then switches to the channel between itself and the next layer and forwards the received packets. For example, a mote in the first relaying layer R, listens on Channel 9 to receive a packet from a sampling layer mote, then switches to Channel 5 to forward it to the next layer, and finally switches back to Channel 9 to listen again. It is important to note that though there are multiple motes in each relaying layer, there is only one base station required. This is because the base station mote has two communication interfaces: the radio to receive the data, and the serial RS232 port on the gateway (MIB510) to forward it. Thus, it has an effective throughput which is already twice of the motes.

D. *Noise Reduction*

At the base station we employ Ephraim-Malah MMSE [6] noise reduction, to remove the background noise. This brings about considerable improvement in sound quality.

VI. Implementation

A. *System Architecture*

The architecture of the system is shown in Figure 3. The system is composed of three layers: the sampling (recording) layer, the intermediate (multi-hop) relaying layers and the receiver (base station) layer. The receiving layer comprises only one mote (the base station), which is connected to a computer using the gateway (MIB510). Currently three motes make up the sampling layer. Each sub-layer of relays also has

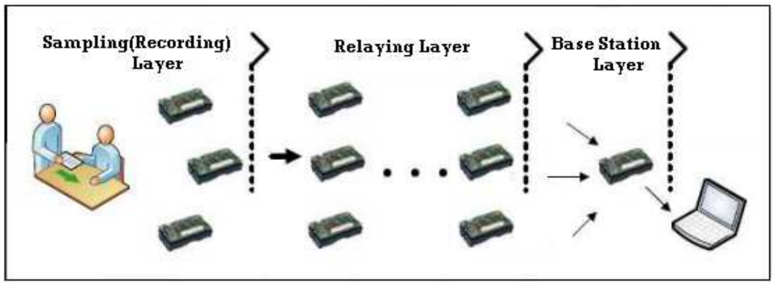

Fig. 3. System Architecture

three motes - the number of these routing sub-layers determine the multi-hop range of the system. We currently deploy two sub-layers in our prototype.

B. Control Flow

In this section, we describe the procedure's control flow. A time division based system allows each sampling layer mote to complete recording, compression and transmission. Each mote gets a fixed quantum of time, in which it only records speech. At the end of its quantum, it sends a *SampleRequest* message, which triggers the next mote's recording quantum. In this time, the first mote is able to compress and transmit its recorded samples. This synchronized round robin scheduling enables speech to be recorded continuously and relayed back to the base station in real time. A natural question arises here - if the motes are working on fairly fixed schedule, we could have also set their timers to control these operations. The drawback with this system is that it becomes very difficult to merge all the speech parts seamlessly if they are not continuous or some parts are repeated. We could also increase the reliability of the system by using acknowledgements for each of the command messages.

Starting the Recording - Sending a Command to the Motes A Java program on PC side, complete with a GUI interface, is used to send commands to the recording motes via the base station. Parameters to be used - for example the sampling frequency and sampling numbers are specified here. This *SampleRequest* command is transmitted via the routing layers in a multi-hop fashion, finally reaching and activating the recording motes. After sending out the *SampleRequest* command, we activate the Serial Forwarder program, which listens to the serial port and waits for the data to arrive on the base station.

Sampling On receiving the *SampleRequest* command, the first mote (say M1) shuts down its radio and starts recording, using the parameters specified. At the end of its sampling quantum, M1 sends the *SampleRequest* command to M2, who starts sampling. M1 in the mean while compresses its data and transmits it to the immediate relaying layer.

Compression and Transmission After send a *SampleRequest* command to M2, M1 starts compression. The Compression module takes the original data log as input to the MRC algorithm described earlier and outputs the compressed data log. This compression takes at most 4 seconds. After compression, M1 then switches to the channel of the nearest relaying layer and transmits the compressed data. The relaying layer simply forwards the data to the next layer.

Starting A New Round Each mote samples for 12 seconds, takes about 4 seconds to compress the data and about 10 seconds to transmit it. After this, it switches to a waiting mode. M1, as an example, finishes its 'work' and waits for a *SampleRequest* command from M3, signaling the start of a new round.

The complete process can be halted using the *Stop* command, given from the base station.

C. Parameter Settings

Sampling and Transmission Setting We set the sampling rate to 5 kHz. Each mote records 60,000 samples in each "round", and logs these into the flash, for compression and transmission. As detailed in Section 5.4 we optimized the values of the packet size and the CSMA/CA random back-off delay, to achieve the very high required data rate. Two experiments were performed to help determine these parameters and verify out solutions. The first one was aimed at finding the optimal packet size, which as shown in Table 2, comes out to be 240 bytes. Fixing the packet size, we now varied the random back-off delay and as shown in Table 3, determined that the data rate is highest when it is 1$\sim$32 bytes. This combination achieves a data rate of nearly 15 kbps, which is very close to the data sheet maximum of 19.2 kbps, and a significant improvement over the typically achieved 4 $\sim$ 5 kbps. We do not increase the data length any further because of two reasons. Firstly, the default header of TinyOS allows only a maximum size of 256 for the data length. Secondly, the higher the packet size, the more is the Bit Error Rate. Thus, we keep it at 240, which gives us the required performance.

Channel Switching Setting Currently, there are 4 layers in the whole system. With FDMA between each, there are 4 channels. Each of them has their own channel. We adjust the frequencies through the interface CC1000Control.

We observe that the frequency synthesizer's performance varies in its working range. In other words, some channels are less reliable than others, thus resulting in packet loss. We experimentally determined and used the channels most suited for reliable high data rate transmission. Figure 4 shows the relation between packet loss rate and channel ID. The channel ID and their frequency values are shown in Table 4.

TABLE II

DATA LENGTH AND TRANSMISSION RATE (BACK-OFF DELAY: 1∼32)

Data Length(Bytes)	Transmission Rate(kbps)
240	14.8
210	14.4
180	13.8
150	13.2
120	12.2
90	11.1
60	9.1
29(default)	5.9

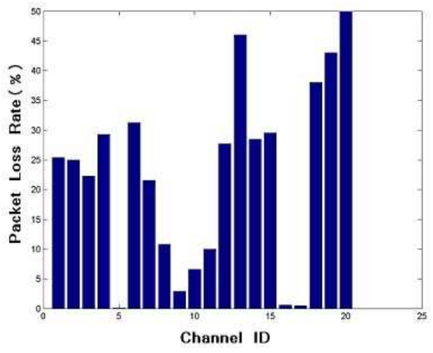

Fig. 4. Channel Test on Packet Loss Rate

Channels 5, 16, and 17 are obviously better than the others and thus we pick those three along with the default MICA 2 channel 9. We note that Channel 9's performance is a little worse than others and thus employ it at the sampling layer, where it is only used to receive the command messages. To scale the system with more hops, these channels could be re-used with SDMA (Space Division Multiple Access).

VII. PERFORMANCE ANALYSIS

We evaluated the performance of our system with respect to several parameters. To quantify the quality of recorded speech we use a full input-output objective measure - the Perceptual Evaluation of Speech Quality (PESQ) [8]. This widely accepted standard is specified in ITU-T recommendation P.862 and is typically used to evaluate perceptual voice quality in telecommunications. The algorithm is implemented in C, takes as input the original speech file and the recorded speech file, and outputs a Mean Opinion Score out of a maximum 5.

It is important to note that the recorded file has to be pre-processed before running the PESQ algorithm. This is primarily to minimize synchronization errors which are caused mainly due to the poor microphone (introduces "stretching of sound") and the occasional packet losses.

A. Compression and Noise Reduction

Compression and noise are two major sources of sound quality degradation. All sample values greater than +3 or lesser than -3, are clipped to 3 or -3 respectively. This distortion is shown in Figure 5. However, the reduction in quality is tolerable and not drastic, since we observe that quality up to PESQ 1.0 is intelligible.

TABLE III

BACK-OFF DELAY AND TRANSMISSION RATE (DATA LENGTH: 240 BYTES)

Back-off Delay(Bytes)	Transmission Rate(kbps)
1∼32	14.8
33∼64	13.4
65∼96	12.2
97∼128	11.3
129∼160	10.4
161∼192	9.7
193∼224	9.1
225∼256	8.5
247∼374	8.1

TABLE IV

CHANNEL ID / CHANNEL FREQUENCY(MHZ)

1/906.704	2/907.838	3/908.715	4/909.653
5/910.391	6/911.396	7/912.234	8/913.156
9/914.077	10/914.998	11/915.920	12/916.758
13/917.763	14/918.614	15/919.439	16/920.397
17/921.450	18/922.187	19/923.088	20/924.083

Figure 6 shows the improvement in speech quality achieved by the noise reduction algorithm - Ephraim-Malah MMSE [6]. Since, this is primarily intended to remove background noise it does not reflect accurately upon the PESQ Mean Opinion Score, because the human ear can "perceive" intelligible speech despite background noise. However some actual listening tests (subjective tests) show that the quality is improved significantly.

B. Distance

Next, we wish to evaluate if the distance of recording motes from the speaker plays a significant role in the speech quality. This is a natural concern, for we wish to optimize the placement of the recording layer of motes with respect to the speaker.

From Figure 7, we can see that the quality first increases and then drops as the distance increases. The increase within two feet of the sound source is relevant to our compression algorithm. When the distance is small, the sound is relatively "louder" to the microphone; thus, the amplitude of the wave is higher. Since our algorithm truncates samples with large values, it induces more quality loss. The decrease beyond this

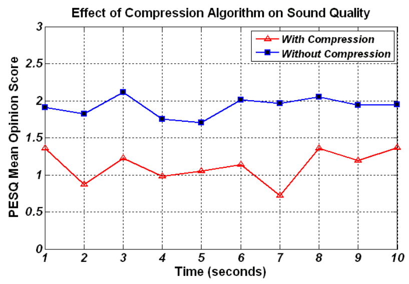

Fig. 5. Effect of Compression Algorithm on PESQ

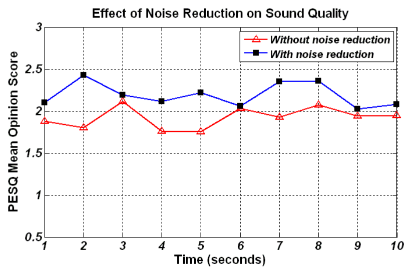

Fig. 6. Effect of Noise Reduction Algorithm on PESQ

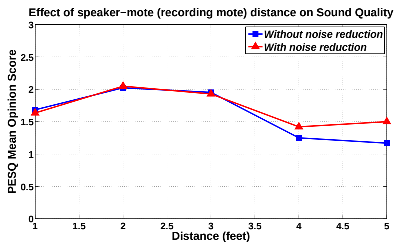

Fig. 7. Effect of Distance on PESQ

two feet distance is expected because sound strength falls with distance. Also, since background noise increases with distance, we note that the noise reduction algorithm is more effective in these scenarios.

C. Multi-hop and Channel Switching

In a multi-hop system, the packet loss rate is an important factor. Besides more chances of collisions, the channel switching also adds to the possibility of packet losses. For each mote, the packet loss rate observed is normally less than 0.5 %. We also observe that the global packet loss rate is normally less than 5%. Due to occasional interference, sometimes the packet loss rate can be as big as 30%. In Figure 8, we can see that the multi-hop system is has a lower sound quality as compared to the one-hop system, but this is not much.

VIII. Discussion

A. Limitation of Our System

Though intelligible, the recorded speech is not perfectly clear. This is mainly because of the 5 kHz sampling frequency used. Typically telephony applications use at least 8 kHz, in compliance with the Nyquist Theorem, failing which we see aliasing, stretching and degradation of speech quality. Other factors are the fact that the microphone was not designed for recording purposes, and if the speech is not loud enough (or motes are further than 4 feet off), background noise causes significant degradation in quality despite the noise reduction.

Packet losses are also a considerable limitation of the system. Since each packet is 240 bytes after compression, it has on an average 600 samples. This implies that packets lost due to interference or collision causes large gaps in speech. Further, it makes synchronization also difficult with respect to applying objective speech quality measures. This packet loss rate has to be tolerated primarily because the system's working is possible only if the motes work at their full capacity in terms of transmission bandwidth, thus leaving almost no room for implementing reliable communication protocols.

The range of the motes, i.e., the distance between each layer is also limited. Because of the rapid and continuous channel switching, our radio communications are not very robust. For reliable results, we need to use almost line of sight always. The packet losses are seen to increase to nearly 30 % when the distance is increased to about 20 feet with walls also attenuating the signal.

B. Future Work

Firstly, we are looking into making the transmission protocol more reliable. It is a non-trivial task because of the already full bandwidth load of the user data itself. However, this could help us reduce the packet loss, and hence improve sound quality.

Secondly, though our main purpose was to demonstrate the serious privacy threat arising out of sensor networks, this research also gives insight into high sampling frequency and data rate applications. We wish to explore this further, possibly towards image and video capturing, and intelligence in the form of in-network processing.

Thirdly, the current implementation of SensorEar is on the MICA2 mote, which is the most basic and cheapest platform in the market. We can expect dramatic improvement in sound quality and radio range, if we port the code onto MICAZ motes. Further, this allows us to explore other system architectures such as coordinated sampling layers, which might achieve higher sampling frequencies. With better processing power, it might be possible to run the noise reduction algorithms on the motes itself, or run more powerful compression algorithms.

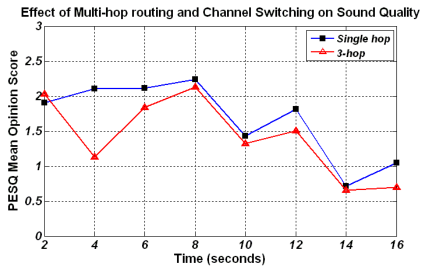

Fig. 8. Effects of Multi-hop and Channel Switching on PESQ

IX. Conclusion

This research work demonstrated a serious threat to an individual's privacy arising out of the widespread use of sensor networks, in the form of a real-time eavesdropping system. We built a complete prototype using commercial off-the-shelf components, which can record and compress speech and transmit it over a multi-hop network. After noise reduction, we were able to hear, with sufficient clarity, the person's conversation. We addressed multiple challenges arising out of the high sampling rate and the severe platform limitations, by designing solutions including a specific compression algorithm, unique system architecture and optimization of the radio stack. The performance evaluation showed that it is a reliable and robust system, and gave speech at a fairly good quality.

Acknowledgement: We thank the reviewers for their valuable comments. This work was supported in part by grant CAREER NSF-0643906.

REFERENCES

[1] ATMEL Co., Data sheet of Atmel128L, `http://www.atmel.com/atmel/acrobat/doc2467.pdf`.

[2] M. Connolly and O. Fergus. Sensor Network and Food Industry. `http://www.sics.se/realwsn05/papers/connolly05sensor.pdf`.

[3] Crossbow Inc, Data sheet of MICA2. `http://www.xbow.com/products/Productpdffiles/Wirelesspdf/MICA2Datasheet.pdf`.

[4] Crossbow Inc, MPR-MIB Series Users Manual. `http://www.xbow.com/Support/Support_pdf_files/MPR-MIB_Series_Users_Manual.pdf`.

[5] J. Daniel, N. Davey, and T. Rice. An accelerometer based sensor platform for insitu elite athlete performance analysis. IEEE Sensors (2004).

[6] M. Ephraim. Speech Enhancement Using MMSE Short-Time Spectral Amplitude Estimator. IEEETrans.on ASSP, vol.32, N6, Decemeber (1984).

[7] J. Han, A. Jain, M. Luk, and A. Perrig. Don't Sweat Your Privacy. Proceedings of 5th International Workshop on Privacy in UbiComp (UbiPriv'07), September (2007).

[8] ITU,PESQ, `http://www.itu.int/rec/T-REC-P.862/en`.

[9] L. Luo, Q.Cao, C. Huang, J. Stankovic, and M. Ward. EnviroMic: Towards Cooperative Storage and Retrieval in Audio Sensor Networks. 27th International Conference on Distributed Computing Systems (ICDCS '07) (2007).

[10] A. Mainwaring *et. al*. Wireless Sensor Networks for Habitat Monitoring. WSNA'02, September 28, Atlanta, Georgia, USA (2002).

[11] R. Mangharam, A. Rowe and R. Rajkumar. Voice over Sensor Networks. 27th IEEE Real-Time Systems Symposium (RTSS), Rio de Janeiro, Brazil, December 2006.

[12] Pulse-code Modulation, `http://en.wikipedia.org/wiki/Pulse-code_modulation`.

[13] K. Pister *et al*. Smart dust: Autonomous sensing and communication in a cubic millimeter. `http://robotics.eecs.berkeley.edu/~pister/SmartDust/`.

[14] J. Princen, A. Johnson, and A. Bradley. Subband/transform coding using filter bank designs based on time domain aliasing cancellation. IEEE Proc. Intl. Conf. on Acoustics, Speech, and Signal Processing (ICASSP) 12, 2161-2164 (1987).

[15] V. Shnayder, B. Chen, K. Lorincz, R. Thaddeus, and M. Welsh. Sensor Networks for Medical Care. Technical Report TR-08-05, Division of Engineering and Applied Sciences, Harvard University (2005).

[16] G. Simon, *et al.* Sensor Network-Based Countersniper System. ACM SenSys, 4, November 3-5, Baltimore, Maryland, USA (2004).

[17] S. Smith. Chapter 27: "Data Compression" of The Scientist and Engineer's Guide to Digital Signal Processing. `http://www.dspguide.com/`.

[18] TinyOS Forum, `http://www.tinyos.net/`.

[19] Wikipedia: Audio Freqency. `http://en.wikipedia.org/wiki/Audiofrequency`.

[20] Wikipedia:Adapative MultiRate Compression. `http://en.wikipedia.org/wiki/Adaptive_multi-rate_compression`.

[21] Wikipedia: FS-1015. `http://en.wikipedia.org/wiki/FS-1015`.