Real-time On-Demand Multi-Hop Audio Streaming with Low-Resource Sensor Motes

Congduc Pham
University of Pau, LIUPPA Laboratory
congduc.pham@univ-pau.fr

Philippe Cousin
Easy Global Market
phlippe.cousin@eglobalmark.com

Arnaud Carer
University of Rennes, IRISA/INRIA
arnaud.carer@irisa.fr

***Abstract*—The european EAR-IT project addresses "real-life" experimentations of intelligent acoustic for supporting high societal value applications in a large-scale smart environment. For instance a city emergency center can request on-demand acoustic data samples for surveillance purposes and management of emergencies. In this paper, we will present experimentations on streaming encoded acoustic on the SmartSantander large scale test-bed. We will present the various audio hardware that were developed to meet the constraints of audio capture and transmission with low-resources devices. We will highlight the main sources of delays and will show how multi-hop streaming of acoustic data can be achieved by carefully taking into account these performance limitations with appropriate audio aggregation techniques.**

***Index Terms*—Smart Cities; Sensor networks; Internet of Thing; Audio streaming; Surveillance**

I. Introduction

There are increasing interests in multimedia contents, such as images and acoustics, for surveillance applications in order to collect richer informations from the physical environment. Multimedia information with small and low-resource infrastructures such as wireless sensor networks (WSN) is quite challenging but the outcome is worth the effort and the range of surveillance applications that can be addressed with WSN will significantly increase and new forms of interactions and decision-making can be implemented.

The EAR-IT project (`www.ear-it.eu`) proposes experimentations of intelligent acoustics for delivering new innovative societal range of services on a large scale. The main targeted applications are smart-buildings and smart-cities where one scenario is an on-demand acoustic data streaming feature for surveillance systems and management of emergencies. The EAR-IT proposed architecture consists of: (i) a limited number of powerful Acoustic Processing Units (APU) with advanced analysis capabilities to accurately detect events of interest and (ii) a large number of low-cost sensing devices, noted IoT (Internet of Things) nodes, that can be used in a complementary way to capture and relay, on an on-demand basis, acoustic data. The scenario assumes that acoustic data will be streamed to a central control system under the supervision of a human operator. The higher density of low-cost IoT nodes compensates their lower audio quality when compared to the specific APUs. They can therefore provide a much higher coverage giving a control center the possibility to monitor larger parts of the city.

In [1], we described our preliminary acoustic experiments with a generic sender mote where encoded audio samples were stored on an SD card and transmitted with an IEEE 802.15.4 device. In this paper, we will present our hardware developments and experimentations to enable real-time multi-hop audio streaming of real-time encoded acoustic data with low-resource devices. There are previous studies on multi-media sensors [2], [3], [4], [5], [6], [7] but few of them really consider timing on realistic hardware constraints for sending/receiving continuous flows of packets. In this paper, we will first highlight the main performance bottlenecks and then present our proposed solutions.

The paper is organized as follows: Section II briefly reviews the EAR-IT test-beds and the various sensor node hardware. Section III presents our audio board development with real-time capture and encoding capabilities. Experimental results of multi-hop acoustic data transmissions on the SmartSantander infrastructure will be presented in Section IV. Conclusions will be given in Section V.

II. The EAR-IT test-beds

The EAR-IT test-beds consist in 2 test-beds: (i) the SmartSantander test-bed and (ii) the HobNet test-bed. The SmartSantander test-bed itself is a 3-location infrastructure project. One main location being the Santander city in north of Spain with more than 2000 nodes deployed across the city. This is the site we will use when referring to the SmartSantander test-bed. The HobNet test-bed is located at MANDAT Intl which is part of the University of Geneva and it is an indoor test-bed. Many information can be found on corresponding project web site (`www.smartsantander.eu` and `www.hobnet-project.eu`).

The Santander test-bed consists in a high number of low-cost Libelium WaspMote boards and a limited number of Libelium Meshlium gateways (see `www.libelium.com`). Most of WaspMote nodes are also repeaters for multi-hops communication to the gateways. The gateways are basically Linux boxes with multiple communication interfaces and have Internet connectivity with a large bandwidth network technology. The WaspMote node is built around an Atmel ATmega1281 micro-controller running at 8MHz with 128KB of flash memory with an XBee 802.15.4 module and one XBee DigiMesh module. In the SmartSantander test-bed, the 802.15.4 module is available for experimentations (mesh

traffic can then be performed with this interface) while the management and service traffic are handled by the Digimesh module. In this paper, we only consider acoustic data transmission/relaying using the 802.15.4 radio module connected to the UART1 of the WaspMote. Most of motes in Santander can reach their gateway in a maximum of 3 hops.

HobNet is a test-bed that focuses on Smart Buildings. Although the HobNet test-bed has several sites, within the EAR-IT project only the UNIGE test-bed at the University of Geneva with TelosB-based motes is concerned. Sensor nodes in the HobNet test-bed consist in AdvanticSys TelosB motes (mainly CM5000 and CM3000) that are themselves based on the TelosB architecture. These motes are built around an TI MSP430 microcontroller with an embedded Texas Instrument CC2420 802.15.4 compatible radio module. The important difference compared to the previous Libelium WaspMote is that the radio module is connected to the microcontroller through an SPI bus instead of a serial UART line which normally would allow for much faster data transfer rates. AdvanticSys motes run under the TinyOS system (`www.tinyos.net`). The last version of TinyOS is 2.1.2 and our tests use this version.

III. Audio board for real-time capture and encoding

A. Audio sampling constraints

At the system level 4KHz or 8KHz periodic 8-bit audio sampling means that the microcontroller must be able to handle 1 byte of raw audio data once every 250us or 125us respectively. Then, when a sufficient number of samples have been buffered, these audio data must be sent while still maintaining the sampling process. Most of IoT nodes are based on low speed microcontroller making simultaneous raw audio sampling and transmission nearly impossible when using only the mote microcontroller. To solve these performance issues, one common approach is to dedicate one of the 2 tasks to another microcontroller: (1) use another microcontroller to perform all the transmission operations (memory copies and buffering, frame formatting, ...) or (2) use another microcontroller to perform the sampling operations (generates interruptions, reads analog input, performs A/D conversion and possibly encodes the raw audio data). With the hardware platforms used in the EAR-IT project we can investigate these 2 solutions :

1) Libelium WaspMote uses an XBee radio module which has an embedded internal microcontroller that is capable of handling all the sending operations when running in so-called transparent mode (serial line replacement mode);
2) Develop a daughter audio board for the AdvanticSys TelosB mote that will perform the periodic sampling, encode the raw audio data with a given audio codec and fill in a buffer that will be periodically read by the host microcontroller, i.e. the TelosB MSP430.

Solution 1 has been experimented and we successfully sampled at 8KHz to generate a 64000bps raw audio stream which is handled transparently by an XBee module running in transparent mode. Although interesting this solution is quite limited because the transparent mode does not allow for dynamic destination address configuration making multi-hop transmission nearly impossible and, more importantly, makes a high usage of the radio bandwidth. Therefore we will describe in this paper solution 2 using the AdvanticSys mote as the host board and we will present experimentation of multi-hop transmission using both Libelium WaspMote and AdvanticSys TelosB motes as relay nodes.

B. Audio board design

The audio board will have its own microcontroller and will handle the sampling operations and will encode in real-time the raw audio data into Speex codec (`www.speex.org`). 8KHz sampling and 16-bits samples will be used to produce an optimized 8kbps encoded Speex audio stream (speex encoding library is provided by Microchip). This audio board is designed and developed through a collaboration with IRISA/CAIRN research team and Feichter Electronics company (`www.feichter-electronics.com`).

The audio board has a built-in omnidirectional MEMs microphone (ADMP404 from Analog Devices) but an external microphone can also be connected. The microphone signal output is amplified, digitized and filtered with the WM8940 audio codec. The audio board is built around a 16-bit Microchip dsPIC33EP512 microcontroller clocked at 47.5 MHz that offers enough processing power to encode the audio data in real-time.

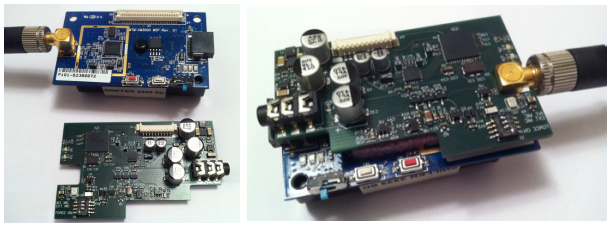

Fig. 1. TelosB with the audio board

From the system perspective, the audio board sends the audio encoded data stream to the host microcontroller through an UART component. The host mote will periodically read the encoded data to periodically get fixed size encoded data packets that will be transmitted wirelessly through the communication stack. Fig. 1 shows the AdvanticSys mote with the developed audio board. The speex codec at 8kbps works with 20ms audio frames: every 20ms 160 samples of raw audio data are sent to the speex encoder to produce a 20-bytes audio packet that will be sent to the host microcontroller through an UART line. These 20 bytes will be read by the host microcontroller and 4 framing bytes are added to the audio data. The first two framing bytes will be used by the receiver to recognize an audio packet. Then sequence number can be used to detect packet losses. The last framing byte stores the audio payload size (in our case it is always 20 bytes).

IV. Experimentations

A. Experimental test-bed

The experiment uses 1 audio source mote consisting of an AdvanticSys TelosB mote with the developed audio board, relay nodes consisting of both Libelium WaspMote and AdvanticSys motes and a receiver consisting of an AdvanticSys TelosB mote connected to a Linux computer to serve as a radio gateway.

The audio source can be controlled wirelessly with 3 commands: ”D” command defines the next hop address, ”C” command controls the audio board power (off/on) and ”A” command defines the audio frame aggregation level which will be described later on. The relay nodes can also be controlled wirelessly and they mainly accept the ”D” command to define the next hop address. The receiver will get audio packets from the AdvanticSys radio gateway, check for the framing bytes and feed the speex audio decoder with the encoded audio data. The audio decoder will produce a raw audio stream that can be played in real-time with `play` or stored in a file by using standard Unix redirection command. A play-out buffer threshold can be specified for `play` to compensate for variable packet jitter at the cost of higher play-out latencies.

We selected a location in Santander near the marina, see Fig. 2(left), to install the audio source and the relay nodes on the same street lamps than the one deployed by the Santander test-bed, see Fig. 2(right). We did not perform tests on the HobNet test-bed yet, but we use both HobNet (AdvanticSys TelosB) and Santander (Libelium WaspMote) hardware as relay nodes.

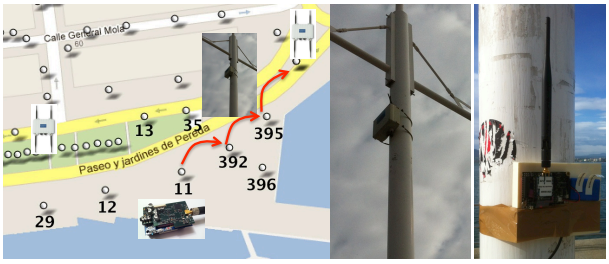

Fig. 2. Test of acoustic data streaming: topology

We placed our nodes on the street lamps indicated in Fig. 2(left), at locations 11, 392, 395 and the top-right gateway. The audio node is on location 11, the receiver is at the top-right gateway location and the 2 relay nodes are at location 392 and 395. With 2 relay nodes, the number of hops is 3. Most of IoT nodes deployed in Santander can reach their associated gateway in a maximum of 3 hops. The original IoT nodes of the Santander test-bed are placed on street lamp as shown in Fig. 2(left). We strapped our nodes as depicted by Fig. 2(right).

In all the tests described here, the transmission power is set to the maximum radio module power (on the CC2420 of the AdvanticSys TelosB, TinyOS sets the transmission power by default to 0dBm) or to the maximum allowed transmission power (in the case of XBee Pro module for instance on the Libelium WaspMote the European regulation sets the maximum transmission power to 10dBm).

B. Packet inter-arrival time & packet loss rate

We use the `wireshark` packet analyzer with a promiscuous sniffer mote to obtain packet statistics. First, the sniffer is placed at location 392 and we observed no packet losses. Second, the sniffer is placed at the top-right gateway and Fig. 3 shows the packet inter-arrival time captured at that location for an audio capture of about 1 minute. In total there were 3040 packets of 24 bytes. We disabled MAC level retransmission to measure the packet loss rate when no reliability mechanism is added.

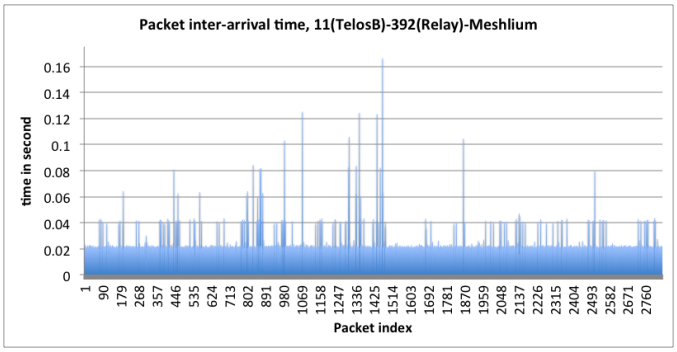

Fig. 3. Packet inter-arrival time

From location 11 to the top-right gateway, we observed 176 lost packets. The packet loss rate is about 5.78%. On Fig. 3, most of packets have an inter-arrival time of about 20ms and packet losses can be clearly identified with higher inter-arrival time. The vertical graduations on the graph can indicate the number of packets that have been lost: the graduation at 0.04 indicates 1 lost packet, the next graduation indicates 2 lost packets, etc. Here, the maximum number of lost packets in a burst is 7 (packet index 1466).

C. Multi-hop: using AdvanticSys relay nodes

We have performed a number of benchmarks to determine for each sensor platform the minimum relaying time when the payload is varied.

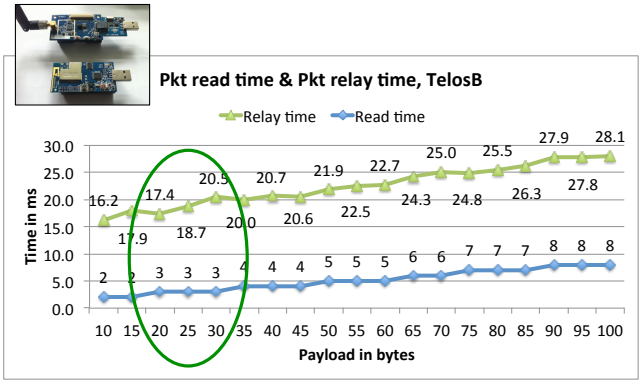

Fig. 4. AdvanticSys TelosB mean relaying performance

We experimentally measured the packet read time (the time needed to read a packet into the application memory space)

and the relay time under static routing. We can see in Fig. 4 that on average an AdvanticSys TelosB relay node needs about 19ms to relay a 25-byte packet. However, as shown in Fig. 5 with packet index from 60 to 80, we can see that sometimes relaying can take more than 20ms. As the audio source sends a 24-byte packet once every 20ms, it may happen that some packets are dropped at the relay node. We observed packet loss rates between 10% and 15% at the receiver.

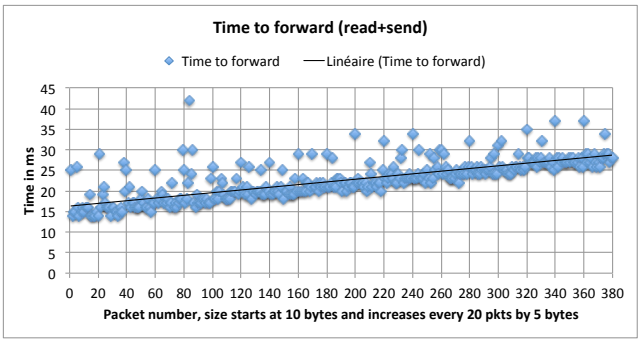

Fig. 5. AdvanticSys TelosB relaying performance raw measures

In order to reduce the packet drop rate, we can aggregate 2 audio packets at the source to provide a 40ms time window for the relaying nodes. In this case, the radio packet payload is 48 bytes and the average relaying time is about 22ms as shown in Fig. 4. Fig. 5 also shows that most of the relaying time are below 30ms (packet index from 160 to 180) therefore avoiding the relay nodes to queue incoming packets. With this strategy, we observe a packet loss rate near to 0. This is due to very little intra-path interferences because the audio source only sends a packet once every 40ms while the contention on the radio channel is in the order of a few ms.

D. Using WaspMote relay nodes

Fig. 6 shows for the WaspMote both packet read time and the relaying time as the packet size is varied. In all our experimentations, for baud rates of 38400, 125000 and 250000, t_{read} remains constant and depends only on the data size. The reason why t_{read} only depends on the data size, at least at the application level, is as follows: most of communication APIs use a system-level receive buffer and when a packet arrives at the radio, a hardware interrupt is raised and appropriate callback functions are used to fill in the receive buffer that will be read later on by the application. Therefore, the baud rate has only an impact on the time needed to transfer data from the radio module to the receive buffer. The time needed to transfer the data from the receive buffer to the application depends on the speed of memory copy operations, therefore depending mainly on the frequency used to operate the sensor board and the data bus speed. As we can see in Fig. 6, t_{read} on the WaspMote is about 20ms and t_{relay} is about 60ms for a 25-byte packet.

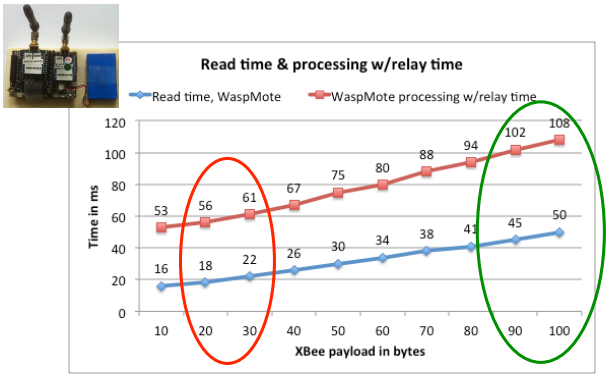

Fig. 6. Libelium WaspMote relaying performance

Therefore the issue we have here is that the source node is sending at the rate of a 24-byte packet every 20ms while the WaspMote relay node needs about 60ms to relay it. We observed packet drop rates above 70% that dramatically degrade the audio quality. With a maximum payload of 100 bytes, the maximum packet aggregation level is 4 (giving a 96-byte radio packet). However, aggregating 4 audio packets only gives a time window of 80ms which is still not enough for the WaspMote as Fig. 6 shows that a minimum of 108ms is needed to relay a 100-byte packet. In this case, we have no other choice than dropping audio packets at the source node: 6 audio packets will be captured providing a time window of 120ms but only 4 will be aggregated in a radio packet, giving a relay time of about 108ms. The extra 12ms give enough time for the relay node of limit packet drops. With this strategy, the initial packet drop rate at the source is $2/6 = 33\%$. The radio transmission has a packet drop rate close to 0 and always less than 5%. Once again, the intra-path interference is very small. The final packet drop rate is on average between 33% and 35%.

E. Audio quality

In order to measure the receiving audio quality, we use the ITU-T P.862 PESQ software suite for narrowband audio to get an audio quality indicator (MOS-LQO) between the original audio data and the received audio data. Fig. 7 shows for various packet loss rates the MOS-LQO indicator value when each radio packet is 20 bytes long, i.e. 1 audio packet in 1 radio packet. The first vertical bar (at 4.308) is the MOS-LQO value when comparing the speex encoded audio data to the uncompressed audio format[1]. It is usually admitted that a MOS-LQO of at least 2.6 is of reasonably good quality. When there is a packet loss, it is possible to detect it by the gap in the sequence number and use the appropriate speex decoder mode. The red bars indicates the MOS-LQO values when packet losses are detected. Without the packet loss detection feature, missing packets are simply ignored and the speex decoder will simple decode the flow of available received packets. We can see that it is always better to detect packet losses.

[1]Reader can listen at the various audio files at `web.univ-pau.fr/~cpham/SmartSantanderSample/speex`

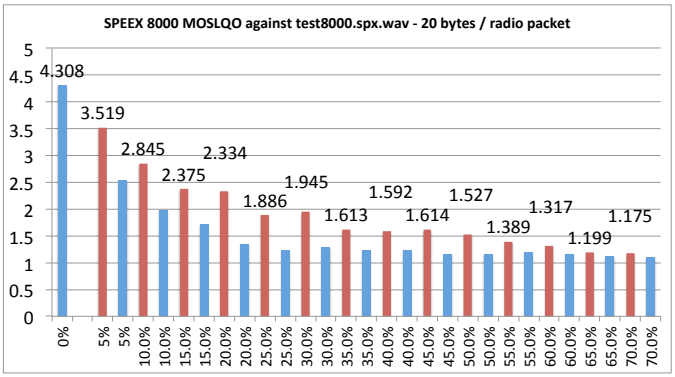

Fig. 7. Audio quality with 1 audio packet in 1 radio packet when packet loss rate is varied

In Fig. 7 we can see that the previous case with an AdvanticSys relay node without audio packet aggregation (between 10% and 15% packet loss rate) still has an acceptable MOS-LQO value. Using aggregation (2 audio packets in 1 radio packet) makes the packet loss rate to be below 5% and therefore provides a good audio quality as indicated in Fig. 8.

When using Libelium WaspMote as relay nodes, our specific aggregation strategy for this hardware platform introduces at the source node a packet drop rate of 33%. Fig. 7 shows that in this case, without additional packet losses in the radio transmission, we can achieve a MOS-LQO value between 1.61 and 1.94 in the best case. However, the audio file is still very understandable for a human operator.

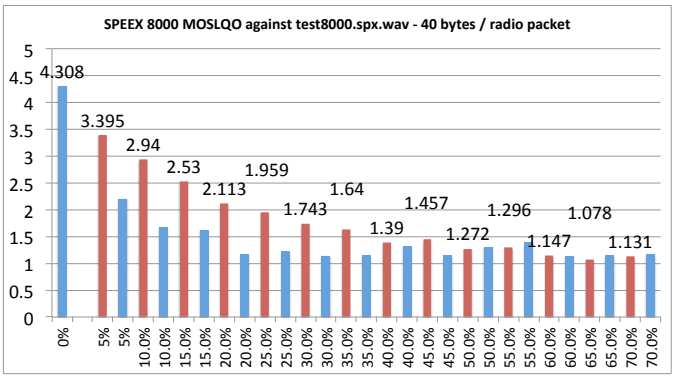

Fig. 8. Audio quality with 2 audio packets in 1 radio packet when packet loss rate is varied

F. Energy consumption of the audio source mote

We also investigated the energy consumption of the audio source node with the developed audio board. Fig. 9 shows the cumulated energy consumption. The first part of the figure shows the idle period where the audio board is powered off and the radio module is in active state. Then, starting at time 43s, the audio board is powered on to capture and encode in real-time during about 20s. The audio packets are sent wirelessly.

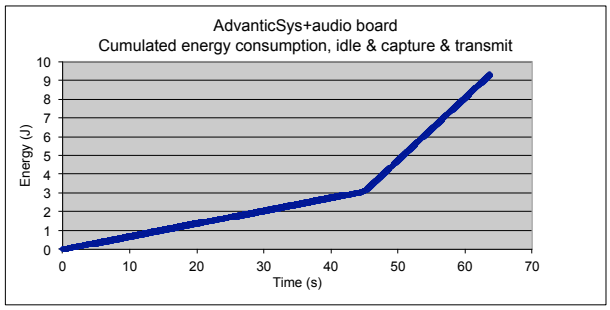

Fig. 9. Cumulated energy consumption

During idle period, the consumed energy is about 0.068J/s (68mW). During audio capture with the radio sending, the consumed energy is about 0.33J/s (330mW). With a 2 AA-battery that approximately have an energy of 18700J, we could continuously capture and transmit during more than 15 hours! Therefore periodic audio streaming scenarios are very possible in the context of smart cities where most of sensor nodes can usually be recharged at night.

V. Conclusions

In this paper, we presented experimentations on the various EAR-IT test-beds for real-time acoustic data streaming. We developed an audio board to sample an encode in real-time acoustic data. The audio board can be plugged on IoT nodes enabling real-time on-demand acoustic data scenario. Prior to the streaming experimentation itself, we first qualified the EAR-IT hardware and highlighted the main sources of delays. We showed that there are incompressible delays due to hardware constraints and software API that limit the relaying throughput. These constraints can be alleviated with appropriate audio frame aggregation strategies and the experiments we performed with the speex codec demonstrated that streaming acoustic data is feasible on Smart Cities infrastructures with reasonably high audio quality and sensor node lifetime.

Acknowledgment

This work is partially supported by the EU FP7 EAR-IT project and the Aquitaine-Aragon OMNIDATA project.

References

[1] C. Pham and P. Cousin, “Streaming the sound of smart cities: Experimentations on the smartsantander test-bed,” in *IEEE International Conference on Internet of Things (iThings2013)*, 2013.

[2] R. Mangharam, A. Rowe, R. Rajkumar, and R. Suzuki, “Voice over sensor networks,” in *27th IEEE International of Real-Time Systems Symposium*, 2006.

[3] L. L. et al., “Enviromic: Towards cooperative storage and retrieval in audio sensor networks,” in *IEEE ICDCS*, 2007.

[4] S. Misra, M. Reisslein, and G. Xue, “A survey of multimedia streaming in wireless sensor networks,” *IEEE Communications Surveys & Tutorials*, vol. 10, 2008.

[5] D. Brunelli, M. Maggiorotti, and F. L. Bellifemine, “Analysis of audio streaming capability of zigbee networks,” in *Proceedings of Fifth European Workshop on Wireless Sensor Network (EWSN2008)*, 2008.

[6] O. Turkes and S. Baydere, “Voice quality analysis in wireless multimedia sensor networks: An experimental study,” in *Proceedings of ISSNIP*, 2011.

[7] E. Touloupis, A. Meliones, and S. Apostolacos, “Implementation and evaluation of a voice codec for zigbee,” in *IEEE ISCC*, June 2011.