Computer Science Department

University of Crete

Fine-grained Localization in Wireless Sensor Networks using Acoustic Sound Transmissions and High Precision Clock Synchronization

Master's Thesis

Evangelos Mangas

November 2008

Heraklion, Greece

University of Crete

Computer Science Department

Fine-grained Localization in Wireless Sensor Networks using Acoustic Sound Transmissions and High Precision Clock Synchronization

Thesis submitted by

Evangelos Mangas

in partial fulfillment of the requirements for the

Master of Science degree in Computer Science

THESIS APPROVAL

Author: _____

Evangelos Mangas

Committee approvals: _____

Angelos Bilas

Associate Professor, Thesis Supervisor

_____

Antonis Argyros

Associate Professor

_____

Panagiotis Tsakalides

Associate Professor

Departmental approval: _____

Panos Trahanias

Professor, Director of Graduate Studies

Heraklion, November 2008

Abstract

Sensors are the fundamental units for ubiquitous computing applications and the key to the popularity of ambient intelligence as an active field of research. Modern sensor architectures do not merely exhibit sensing abilities, they also have increased processing and communication capabilities and most importantly, their volume can be smaller than the size of a coin. Localization is a requirement for most smart applications.

The purpose of our work is to provide a localization method, built to operate on sensors without the need for external infrastructure, excessive hardware or great resource consumption. We achieve localization by having each node produce an audible sound pulse while the rest of the nodes are sensing the audio frequency spectrum. All listeners capture timestamps in a global synchronized timescale at the reception of the sound and we calculate sound time of flight for each one of them by subtracting the sounder's timestamp from each listener's timestamp. We then use sound time of flight measurements so as to estimate distance from the sounder. The focus of our work is (a) on high accuracy clock synchronization and (b) on sound detection for range estimation. In order to provide nodes with synchronized clocks, we have implemented a synchronization protocol operating on MAC-Layer that does not introduce significant communication cost. We demonstrate through simultaneously raising interrupts to the network nodes that our synchronization mechanism guarantees average synchronization precision of $4\mu s$. We also present a technique for efficient sound detection that

does not require excessive resources and results in consistent detection in a range of environments. Our experiments reveal that localization is possible to be performed exclusively by sensors and yield average location estimation error 11cm in distances up to 700cm in certain environments.

Περίληψη

Οι αισθητήρες είναι οι δομικοί λίθοι για τις εφαρμογές πανταχού παρόντος υπολογιστή και το κλειδί της δημοτικότητας της διάχυτης νοημοσύνης ως ενεργό τομέα έρευνας. Οι σύγχρονες αρχιτεκτονικές αισθητήρων δε διαθέτουν απλά δυνατότητες αίσθησης, αλλά επιπλέον έχουν επαυξημένες ικανότητες επεξεργασίας και επικοινωνίας και κυρίως, το μεγεθός τους μπορεί να είναι μικρότερο από αυτό ενός κέρματος. Η δυνατότητα εντοπισμού θέσης αποτελεί απαίτηση για την πλειονότητα των έξυπνων εφαρμογών.

Σκοπός της παρούσης εργασίας είναι να παρέχουμε μια μέθοδο εντοπισμού θέσης, κατασκευασμένης ώστε να λειτουργεί σε αισθητήρες χωρίς απαίτηση για ύπαρξη εξωτερικών υποδομών, περίσσιου υλικού ή μεγάλης κατανάλωσης πόρων. Προτείνουμε εντοπισμό θέσης με την παραγωγή ηχητικού παλμού κάθε φορά από έναν κόμβο του δικτύου, με παράλληλη ρύθμιση των υπολοίπων κόμβων ώστε να ακούν στο φάσμα των ακουστικών συχνοτήτων. Όλοι οι κόμβοι οι οποίοι είναι ρυθμισμένοι να ακούν, καταγράφουν σε μια κοινή, συγχρονισμένη κλίμακα τη χρονική στιγμή της λήψης του ήχου και υπολογίζουμε το χρόνο διάδοσης του ήχου για καθένα από αυτούς αφαιρώντας από το χρόνο λήψης το χρόνο παραγωγής του. Έπειτα χρησιμοποιούμε τις μετρήσεις για το χρόνο διάδοσης του ήχου για να παράγουμε εκτίμηση της απόστασης απο τον κόμβο που τον παρήγαγε. Η εργασία μας εστιάζει (α) στο συγχρονισμό ρολογιών υψηλής ακριβείας και (β) στον εντοπισμό ήχου για εκτίμηση απόστασης. Προκειμένου να εφοδιάσουμε τους κόμβους του δικτύου με συγχρονισμένα ρολόγια, αναπτύξαμε ένα πρωτόκολλο συγχρονισμού στο επίπεδο σύνδεσης δεδομένων

το οποίο δεν εισάγει σημαντικό κόστος επικοινωνίας στο σύστημα. Προκαλώντας ταυτόχρονισμενα διακοπές σε κόμβους του δικτύου αποδεικνύουμε οτι το πρωτόκολλο συγχρονισμού εξασφαλίζει μέσο σφάλμα συγχρονισμού $4\mu s$. Επίσης παρουσιάζουμε μια τεχνική για αποτελεσματικό εντοπισμό ήχου που δεν απαιτεί υπερβολικούς πόρους και επιτυγχάνει εντοπισμό θέσης σε ένα εύρος συνθηκών περιβάλλοντος. Σύμφωνα με τα πειράματά μας ο εντοπισμός θέσης είναι δυνατό να πραγματοποίηθει αποκλειστικά με χρήση αισθητήρων, με μέσο σφάλμα εντοπισμού $11cm$ σε αποστάσεις μέχρι $700cm$ σε συγκεκριμένες συνθήκες περιβάλλοντος.

Acknowledgments

I feel grateful to my supervisor, Angelos Bilas, for his valuable assistance and guidance throughout my studies in the field of Computer Science. Extensive discussions concerning my work, combined with his wide knowledge and his analytical skills have always been of priceless value.

Special thanks to all past and current members of the CARV laboratory that have proved to be great colleagues and friends and especially Michalis Ligerakis, Kostas Kapelonis, Stavros Passas, Stamatis Kavadias, Giorgos Tzenakis, Yannis Kessapidis, Dimitris Papadeas, Yannis Klonatos, Michalis Alvanos, Giorgos Passas and Zoe Sebepou.

I would also like to thank my friends who have always supported me and made this journey towards master degree look like a pleasant trip. Special thanks to the fellows of the club "The fellow-travelers", Themis Dakanalis, Eleytheria and Eleni Spyridakh, Kallia Bathianaki, Chrysostomos and Dimitris Zeginis, for the moments we have all shared together. I would also like to thank Iordanis Aslanidis, Kostas Katertzis and Antonis Papadogiannakis, for being there for me. I could definitely not omit saying how grateful I am to Kostas Agnantis, Zoi Roupakia, Dimitris Tsantilas and the rest of the company of my undergraduate years in Thessaloniki who already follow their own path in life and who I rarely get to see but every time this happens they fuel me up with incredible strength to move on.

Last and most importantly, I would like to thank my beloved family, my parents Dimitris and Stella and my sisters Maria and Apostolia for their constant support and encouragement.

Evangelos Mangas

Heraklion, November 2008

Contents

List of Figures

Chapter 1

Introduction

During the recent years, the field of science named ubiquitous computing has been gaining importance and therefore the attention of the scientific community. Ubiquitous computing is a term used to describe applications that are functioning on a distributed way, on the purpose of facilitating our every day life. Modern sensor architectures do not merely exhibit sensing abilities, they also have increased processing and communication capabilities and most importantly, their volume can be smaller than the size of a coin. Therefore, implementations can be distributed, flexible and more user-targeted than before, allowing for a variety of applications. Sensors nowadays can be used to measure all kinds of effects, from pressure and temperature to acceleration and concentration of chemical combinations in the surrounding environment. Consequently, wireless sensor networks can be used in case of monitoring, either indoor for security, health [11] or industrial reasons [28, 14], or outdoor, in cases of inhospitable habitats or environmental monitoring [5, 17, 6] allowing the absence of otherwise expensive infrastructure. Furthermore, smart house applications make extensive use of wired and wireless sensors [24].

A common place of the vast majority of applications built for wireless sensor networks is the requirement that sensors comprising the network can

calculate their position in space, through a ranging system. This way not only sensor deployment is easier, since no preconfiguration is required, but also applications can be more effective, taking advantage of this extra network feature. This applies well in the case of centralized and decentralized tracking, such as when localizing animal calls in a habitat [29], in structural health monitoring [13, 4], smart house applications where sensors identify the location of persons in house and adjust the environment accordingly so as to save the resident both energy costs and trouble [31] and health applications, when through the sensor network, care providers may monitor residents' health and life habits and watch for chronic pathologies [22]. Furthermore, applications such as robot navigation [21] or even emergency navigation [15, 12, 23], when a target needs to be directed through a region, may also benefit from the function of a ranging system in the sensor network.

Our work focuses on localization; however we want to achieve high accuracy without making use of excessive resources. We rule out the use of extensive external infrastructure that would cost time and money in order to put into use. Besides that, we want the network nodes to manage localization based on their sensing capabilities, without having to add any extra sensing equipment that would be either of big size or would cause great energy consumption. We choose to implement acoustic ranging, as measuring acoustic sound time of flight is less susceptible to interference than RSSI, audible sound is omnidirectional and ranging can be implemented with low-cost hardware that can be embedded on the nodes without posing any extra demand for power supply.

For the purpose of localization, having to decide between measuring Differential Time of Arrival (DToA) or just Time of Arrival (ToA) which requires the nodes to maintain a synchronized clock, we opt to implement the second. In DToA a node should emit two different kind of signals at

once, such as an RF and ultrasound or audible sound signal, so that the RF signal whose speed of transmission is much faster triggers the start of counting at the listener till the sound signal arrives. This way distance is calculated based on the speed of the second signal and the time elapsed. The defect of DToA is twofold: First using local timer's value in calculations may cause erroneous measurements from node to node, if any drift exists between timers and secondly switching into listening for the sound signal right after listening the RF signal can be quite a demanding task, especially when the nodes are close to each other and the time available in which the switch has to be completed is limited. For this reason we choose to apply synchronization and measure ToA of a sound signal. Since the nodes are synchronized beforehand, both the node producing the sound and the one listening to it timestamp it's beginning in a synchronized timescale. All we have to do then is translate the sound time of flight into distance, which is an easy thing to do as the speed of sound is known.

Besides, synchronization is yet another requirement of wireless sensor network applications, regardless of localization. Maintaining a global timescale makes it possible to order all the events sensed by a group of sensors in a common chronological scale, so that monitoring makes sense. For example when sensors are monitoring sounds in a room, or when sensing the diffusion of a chemical combination, it is important to know which measurement came first. Moreover, timestamping on a global timescale can be also useful when sending packets for network security purposes, or even when scheduling events like turning off the antenna for a period of time for energy saving reasons. The best way to achieve synchronization is through RF message exchange and given that the cost in resources that communication has over processing, high accuracy in synchronization is desirable even when the need for synchronization of high precision is not imperative. With fine-grained synchronization, resynchronization may take place less often, allowing for

energy savings.

For the purposes of our work we use Mica2dot [1], the coin-sized Berkeley motes that operate on an 8-bit, 3.6MHz AVR processor, communicate through RF at 915MHz and come with sensor boards equipped with various sensing modalities, including a microphone. The operating system running on the sensor motes is CORMOS [32], an event driven runtime environment for wireless sensor networks. Our contributions are:

⇒ First we design and implement a synchronization protocol, that can provide all nodes in the wireless sensor network with a common timescale of high precision. Our objective is to take full advantage of the exact nature of the sensors, that allows low level (MAC-layer) programming, and use its features in order to build a synchronization protocol that guarantees high precision, yet is spare in energy consumption and scalable at the same time. Our protocol has the procedure of calculating transmission delay incorporated into it, in order to eliminate any calibration requirements when devices with different antenna chipsets are used. Unlike most previous work, we achieve scalability and efficiency by having timestamped RF messages broadcasted in the network. We achieve great precision by implementing linear regression on the timestamps received by each node. Our protocol has the advantage of combining high precision along with low communication requirements.

⇒ Then we use our synchronization scheme to design a high accuracy location sensing protocol, that operates using only off-the-shelf, low-cost sensors We use sounders of audible frequency 2.4kHz directly attached to our sensor motes and along with the default microphone of the sensors we timestamp the beginning of sound emissions at sounder and listener and use the timestamps to infer distance. Unlike most techniques proposed so far, all processing is done online by the node. Therefore there is no need for extra hardware or special infrastructure.

Our localization scheme is also scalable, since all nodes may produce or listen to the sound, thus aggregating distances is possible.

The rest of the thesis is organized as follows:

i) In Chapter 2 we describe the design and implementation issues of both our synchronization and localization protocols.

ii) In Chapter 3 we evaluate the efficiency of our synchronization and localization scheme based on experimental results.

iii) In Chapter 4 we discuss the limitations of our protocol.

iv) In Chapter 5 we provide the state of art as far as clock synchronization and localization in sensor networks are concerned.

v) Finally, in Chapter 6 we summarize our work and report our conclusions.

Chapter 2

System Design and Implementation

2.1 Synchronization protocol design

In this section we present our method to achieve Mica2dot mote synchronization. Since synchronization is implemented exclusively through RF message exchange, we first outline the phases that RF communication comprises of, we describe the exact steps we take so as to manage temporary synchronization and then we go on to analyze how that scheme can be implemented for permanent synchronization.

2.1.1 MAC-Layer implementation

Since we have chosen to apply synchronization through RF message exchange, it is of crucial importance to have a clear view of the stages that RF communication comprises of and the degree of determinism of time that it takes each stage to complete as well. As already analyzed in previous work [9, 18], a single action of message exchange can be decomposed into the stages shown in Figure 2.1:

i) Send time: It is the time required for the message to be created and

the send request to be issued to the MAC-Layer. The time it takes this stage to complete is non deterministic. It depends on processor load and can be as high as hundreds of milliseconds.

ii) Access time: It is the amount of time needed till the sender gets access to the sending channel and starts transmitting. This stage is non-deterministic as well. It depends on network load and is in the range of milliseconds.

iii) Transmission time: It corresponds to the time needed for the sender to transmit the message. It is in direct dependence upon the antenna transmission rate and may vary from hundreds of microseconds for a single byte transmission to hundreds of milliseconds for large messages.

iv) Propagation time: It is the time of flight between the sender's and the receiver's antenna. It depends almost exclusively on the distance between the antennas and can be calculated, given that the speed of light is known and equal to 3×10^8 m/s. For a distance of 135m which is the best case transmission range for mica2dot motes [1] it is less than 0.5us.

v) Reception time: The time required for the receiver to receive the message. It has similar properties to the transmission time.

vi) Receive time: It is the amount of time needed for the receiver application to be notified of the received message.

We choose to bypass the uncertainty that poses the non determinism of the send time, access time and receive time by developing a protocol that functions on the MAC-Layer, as close to the transmission phase as possible. This way we only have to focus on transmission, propagation and reception delay. Besides, sensor node architectures and especially Mica2Dot's architecture and their operating systems favor such an approach.

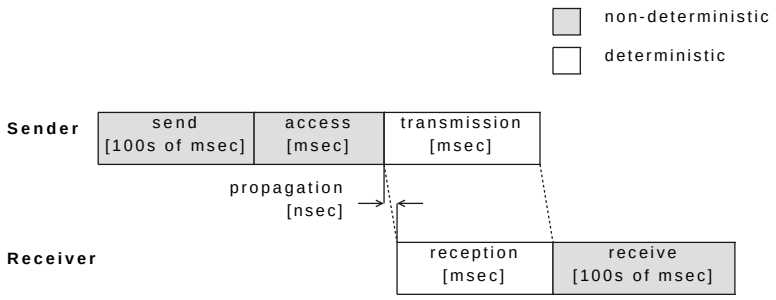


FIGURE 2.1: Stages of message delivery delay over RF.

2.1.2 Timestamp formation

We go on into making a more in depth analysis of the actions that take place in the transmission and the reception phase. To make things simpler, we consider that the data unit to be sent is the smallest possible (i.e. a byte in the case of mica2dot) and suppose that the sender and the receiver are already configured to send and receive correspondingly.

⇒ At the sender, the antenna interface (that would be the serial peripheral interface - SPI) after having forwarded the previous byte into the antenna, raises an interrupt signaling that right after a time delay equal to the interrupt routine service delay the processor will be ready to write the next byte to be sent on the interface buffer.

⇒ The byte gets shifted into the antenna at a rate determined by the antenna transmission rate. At the moment the last bit has been shifted into the antenna a new interrupt is raised so that the processor may write on the interface buffer a next byte to be sent.

⇒ Every bit that enters the antenna needs some time in order to get modulated into electromagnetic waves. The modulation time may vary in different antenna chipsets and configurations (e.g. when a more efficient encoding like Manchester encoding is used), yet for the same

type of antenna chipsets with the same configuration it is quite stable, only sometime introducing some jitter.

⇒ Modulated bits are propagated and then demodulated at the receiver side. The receiver's antenna is shifting demodulated bits into the SPI register.

⇒ As soon as eight consequent bits have been received and shifted into the SPI register, an interrupt is raised and the processor, right after an interrupt routine service delay, collects the received byte and manages it likewise (probably stores it temporarily in a buffer till the whole message is received).

⇒ Most times it happens that the receiver receives bytes in a different alignment than the actual byte alignment in the sender. Therefore the interrupt at the receiver is not raised at the time the actual byte has been received, rather than when the byte plus some extra offset bits (either belonging to the next byte that the sender has sent or being just noise) have been shifted into the SPI register.

As can be seen in figure 2.2, it is essential for us to measure the exact transmission delay - in receiver's local time scale - from the moment that the last bit of a given byte has been shifted into the sender's antenna (timestamp T_A), thus causing an interrupt to be raised, to the moment the last bit of the same byte has been shifted into the SPI register at the receiver (timestamp T_B). The reason is that if d is the aforementioned metric's value, then T_A and $T_B - d$ refer to the exact same moment in absolute time.

For that purpose it is important to get an accurate local timestamp at the sender as soon as the byte is sent and the interrupt routine is getting executed and an accurate local timestamp at the receiver at the moment the actual byte has been received. Consequently, at the receiver side we need to get a local timestamp within the interrupt routine that will be executed right

after the full byte has been received and then subtract the time that was needed for the extra (8 minus offset) bits that have been received because of the misalignment.

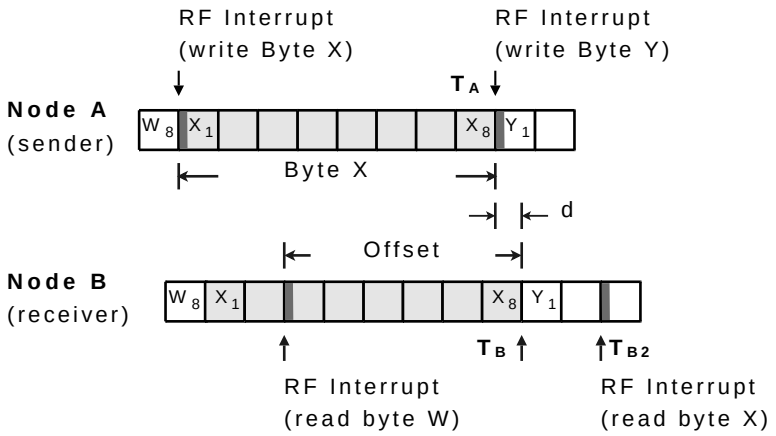


FIGURE 2.2: Transmission and reception timestamping.

The interrupt service delay is in the range of some microseconds and may vary significantly only if another interrupt routine is being executed at the time that the SPI interrupt is raised. However, that uncertainty is something we can handle by capturing multiple timestamps for a succession of bytes at transmission. Thus, we may estimate the amount of time any byte needs to be transmitted at the sender (or accordingly at the receiver) by using a median filter. This way, having available the timestamps of a succession of bytes transmitted before the byte of interest, we can compensate for any delay noticed in the execution of the interrupt routine for the exact byte.

2.1.3 Transmission delay

The main idea for transmission delay calculation can be also found in [20] where NTP is described. Suppose we have two nodes, node A and node B, they only have to exchange timestamps through a symmetric transaction, as shown in Figure 2.3.

The timestamps captured, as described in 2.1.2 will have the property

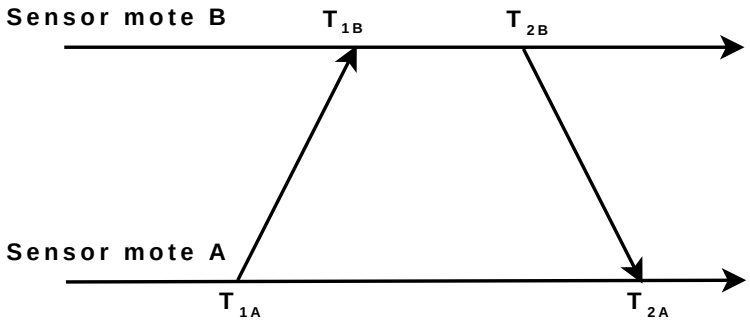


FIGURE 2.3: Round trip time calculation procedure.

that $T_{1A} + d$ and T_{1B} refer to the same moment in absolute time and T_{2B} and $T_{2A} - d$ refer to the same moment in absolute time as well. They can therefore be used in order to calculate the transmission delay by means of the following equation:

$$d = \frac{(T_{2A} - T_{1A}) - (T_{2B} - T_{1B})}{2} \tag{2.1}$$

However, this equation is valid only in the special case when the two local timers are of the exact same frequency, i.e. there exists no skew. This assumption is rarely true and ignoring the clock skew may introduce significant error in calculations. Nevertheless, if we know that T_A clock ticks in node A and T_B clock ticks in node B correspond to the same absolute time period, then a more accurate expression of Equation 2.1 can be formed, by projecting time measured with local timer B to time measured by local timer A:

$$d = \frac{(T_{2A} - T_{1A}) - \frac{T_A}{T_B} \cdot (T_{2B} - T_{1B})}{2} \tag{2.2}$$

2.1.4 Temporary synchronization

Every node in the network at first measures the transmission delay between itself and a master node which it chooses as a synchronized clock timestamps' provider. At first node A has to realize the symmetric two message exchange procedure already described in 2.1.3 so that it can measure the transmission delay d between itself and node B. Once transmission delay has been measured there is no need to repeat the procedure, as long as the same node B is used as timestamps' provider. We have chosen to implement this procedure and not to hardcode the transmission delay measured for mica2dot sensors as in the same environment may be deployed a variety of sensors. This way sensor nodes of different antenna chipsets could also provide synchronization timestamps as long as they transmit in the same frequency and are properly programmed. Right after that, node B will have to broadcast synchronization messages that contain synchronized clock timestamps. Upon receiving a synchronization timestamp T_B, according to what we have already been mentioned in 2.1.2, T_B and $T_A - d$ will refer to the same moment in absolute time. Thus node A can make use of T_B in order to compensate for the offset between the two timers.

As already argued in [8] nodes should not set their clock or discipline their frequency, rather than let it run in its natural rate. It is an optimal choice in wireless sensor networks to have every node maintain a virtual clock, by means of maintaining a set of local and remote timestamps and have it's value calculated on demand, whenever a synchronized timestamp is needed. This way we avoid the continuous calculations needed to have the local timer periodically fixed, we do not cause any problems in scheduling with the local timer going back and forth because of the adjustments and most importantly, we manage a higher accuracy in drift compensation. Conforming to this argument node A does not use the pair of timestamps T_B and $T_A - d$ in order to directly adjust its local timer, rather than stores

this pair of timestamps along with some more and uses it in order to project is local time to synchronized time by means of linear regression.

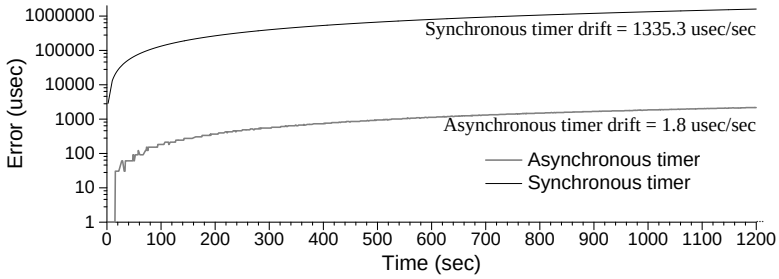
2.1.5 Long term synchronization

Mica2dot devices function on an Atemga128L processor. There are four timers in this processor, two 8-bit timers and two 16-bit timers. The 16-bit timers and one 8-bit timer can only be clocked synchronously at 3.6 MHz (3,686,400 Hz) which is the processor clock frequency and provides a timer tick every 270ns. The other 8-bit timer can either be clocked synchronously or asynchronously by an external crystal of 32,768 Hz frequency, i.e. a timer tick corresponds to 30us.

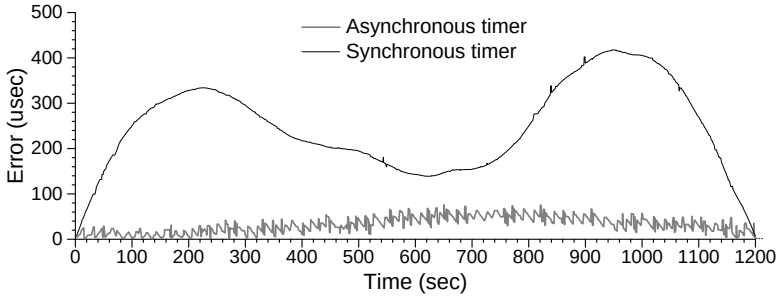
For the purpose of synchronization, we are interested in using a timer that provides the greatest timing granularity on the one hand, so that our measurements have the best accuracy possible and, on the other hand, has either no drift or a drift that is stable in time so that it can be easily compensated for. As far as the timer granularity is concerned, it seems that using the processor clock is an optimal solution. In order to evaluate the drift, however, we had to perform a series of experiments.

Consequently, we simultaneously produce a series of external interrupts to pairs of motes for a period of time. Every mote, at the moment that the execution of the interrupt service routine begins, produces two timestamps; one using the 16-bit timer that is clocked synchronously and another one using the 8-bit timer that is clocked asynchronously. All the timestamps are then collected in a computer, through a gateway mote.

At first we use the timestamps in order to compare the drift both between the synchronous and the asynchronous timers. Supposing that the first timestamps collected by the motes were used so as to synchronize them, we calculate the accumulated error in time in both cases. It comes out that the drift introduced by synchronous timers is up to some milliseconds per second, whereas the drift introduced by asynchronous timers is in the



(a) Accumulated error



(b) Drift compensated error

FIGURE 2.4: Accumulated and drift compensated error between synchronous and asynchronous timers.

range of some microseconds per second. Figure 2.4(a) is an example of the accumulated error calculation in a time period of 1000s.

We next compensate for the drift calculated earlier and check the timing error anew. Apparently, the synchronous timers seem to be less stable in time and introduce great changes in drift compared to the asynchronous ones. That means that the drift compensation is more effective in asynchronous timers. Figure 2.4(b) is the estimated error despite drift compensation, for the same measurements also used in 2.4(a).

Therefore, the optimal choice is a combination of both timers, by using the asynchronous timer in order to adjust a virtual synchronous timer. This way the synchronous timer has accuracy determined by the synchronous timer along with precision determined by the asynchronous timer. Repeat-

ing the experiments, this time having both motes produce raw synchronous timer and adjusted synchronous timer timestamps, we get the results shown in Figure 2.5. It can be seen that the adjusted timer error remains in the scale of some microseconds for long time periods, thus allowing for the synchronization to take place less often and saving communication and computational costs.

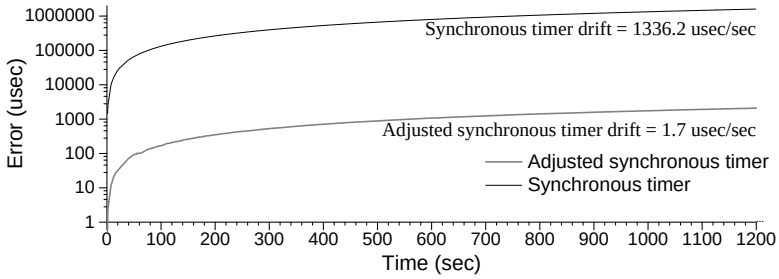
All metrics used in the current work, including transmission delay, are measured using the adjusted synchronous timer. So are the timestamps mentioned in 2.1.4. Every node maintains a set of the four pairs of local and remote timestamps, i.e. the four pairs of timestamps most recently formed, and uses linear regression in order to calculate synchronized time st(T_A) for any given local timestamp T_A. This way we achieve average synchronization error equal to 4us for node B broadcasting synchronization messages every 31s.

2.1.6 Network scheme

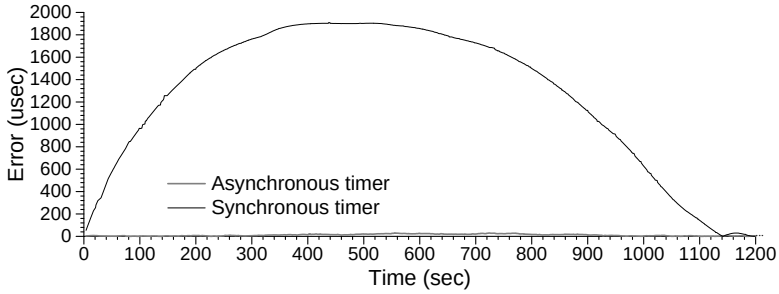
Our current view of the network consists of a single master node which periodically broadcasts timestamps that are to be used by the rest of the sensor motes in order to keep themselves synchronized. This scheme adapts well in small scale wireless sensor networks, given that the antenna transmission range for mica2dot is about 135m, which means that it is not rare that all nodes are within range of a master node. Nevertheless the idea for a multihop implementation exists and is to be implemented in the near future.

2.2 Localization protocol design

Next to building the synchronization implementation, we put it into use, in order to create an application where sensor nodes use sound emissions in order to localize themselves in space in an automated way. For simplicity reasons suppose we have a network of only two synchronized sensor nodes. The principle idea is having one node produce a sound while the other is



(a) Accumulated error



(b) Drift compensated error

FIGURE 2.5: Accumulated and drift compensated error between synchronous and virtual synchronous timers.

sensing the audio frequency spectrum. The node that produces the sound (sounder) creates a timestamp T_S of the moment the sound was produced, while the other mote (listener) creates a timestamp T_L of the moment the sound was sensed. Given that the two timestamps are in a common synchronized scale, then $T_L - T_S$ is the sound time-of-flight (ToF), i.e. the time needed for the sound to travel from the sounder to the listener. The speed of sound in the air is known to be equal to $V_S = 344m/s$. Therefore, we could calculate the distance between the two nodes using the following equation:

$$Dist = V_S \cdot (T_L - T_S) \tag{2.3}$$

Timestamp T_S can be easily captured and refers to the moment that the sounder applies high voltage to the general purpose input/output (GPIO) pin where the buzzer is connected. The most challenging task is to capture timestamp T_L in the listener, so as to to obtain a valid estimation of distance.

The listener, while waiting for the sound, is operating in ADC free running mode, with the ADC converter continuously converting sound measurements that are captured from the microphone into 10-bit values. Every time a new measurement is produced, an interrupt is raised. Our purpose is to be able to recognize the moment that the sound becomes present and produce a timestamp that refers to the beginning of the execution of the ADC interrupt handler which provides the first measurement that we have accepted as indicative of the presence of sound. Given that a new ADC measurement is produced every 52us (208 processor cycles) the maximum feasible accuracy is equal to 1.7cm. In order to put this idea into work the challenge is double:

⇒ The synchronization error should be in any moment less than 32us, so that we can be sure that the timestamp T_L captured will not be misleading as to which ADC interrupt it refers to, making us lose 1.7cm of accuracy, if not even more. This issue has been successfully addressed by our synchronization scheme.

⇒ We have to cope with the uncertainties posed by sound creation, propagation and reception. This is the part of the work that we are to analyze next.

2.2.1 Hardware description

Our objective is the localization implementation to be based on commodity hardware, i.e. off-the-shelf components that are neither expensive nor difficult to be found. This choice was made on the basis that cost is an important factor in deploying ambient intelligence applications. For that purpose, we

use the default microphone that comes with the mica2dot sensor platform, which operates in audible sound frequencies from 20Hz to 16kHz. The microphone is connected to an analog to digital converter that is configured to have a sampling rate of 17.7 kHz. For the given sampling frequency, according to Nyquist theorem the maximum sound frequency we can sample is 8.85 kHz and we definitely need a sound of high enough frequency as sound peak to peak measurements are the ones that are important to identify the existence of sound and the higher the frequency the more such measurements are available. The buzzers that we used in order to produce sound are EMX-7T01SP 1.5V devices, producing a sound of 2.3 KHz frequency. The buzzer is controlled by the INT0 GPIO pin of the sensor.

2.2.2 Calculating amplitude and period

We want the processing of data in the listener to be done online, at the moment of capturing them, so as not to waste processing time and excessive memory for storing all ADC values and the corresponding synchronized timestamps. What we get from the ADC is a series of raw measurements. In case of a sound pulse in 300cm distance the ADC measurements look like those given in Figure 2.6. First we need to reconstruct the signal at the receiver, so that we may know if we have a sound present. Given that the sounder frequency is 2.3kHz whereas the sampling frequency is 17.7kHz, it takes 7 to 8 ADC measurements to sample a single period of the signal. We identify as local maxima L_{MAX} or just peaks of the sound signal the measurements whose value is greater than four previous measurements and greater or equal to four following measurements. We also get to keep the lowest value L_{MIN} between consecutive peaks. This way we are able to calculate the peak-to-peak amplitude of any existing sound as given by equation 2.4.

$$V_{p-p} = L_{MAX} - L_{MIN} \tag{2.4}$$

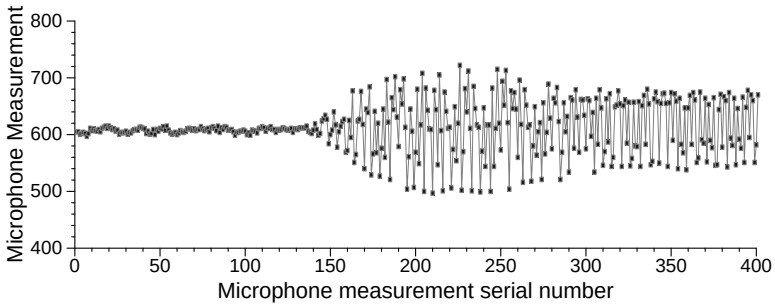
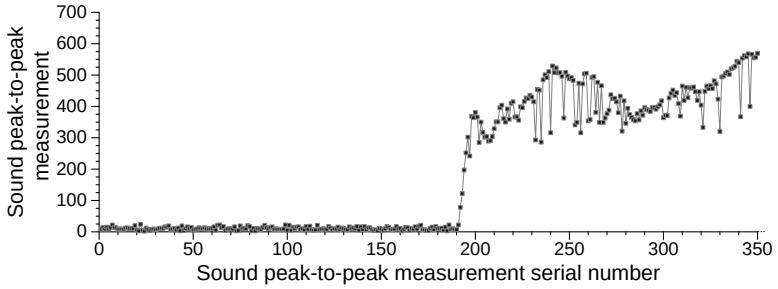


FIGURE 2.6: Raw sound measurements.

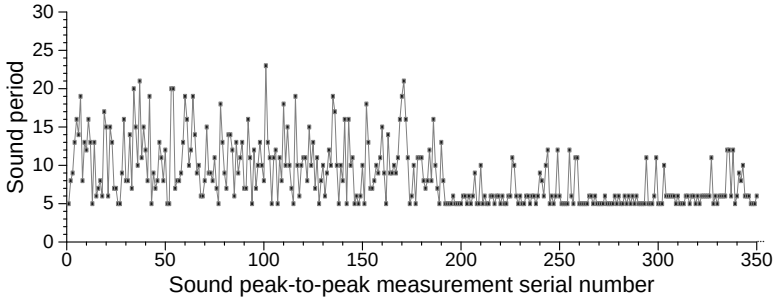
Moreover, the time elapsed between consecutive peaks can be used in order to calculate the period of the sound wave that is being sensed. In figure 2.6 can be seen raw ADC measurements that have been obtained at the beginning of a sound presence with the buzzer being 300 cm away from the buzzer. In figures 2.7, 2.8 and 2.9 are depicted the peak-to-peak amplitude and period of the sound as calculated by the measurements received at the beginning of a sound pulse, with the listener being 100cm, 300cm and 500cm away from the buzzer, but in light of sight. In all cases it is apparent the transition from silence to sound, yet with increasing distance the moment that the sound begins becomes less prominent.

2.2.3 Average filtering

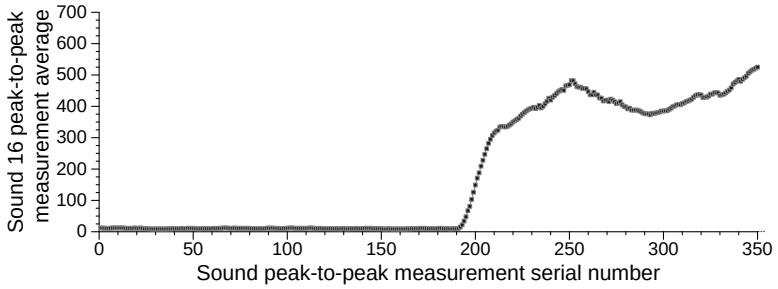
In order to avoid detecting false sounds in room, i.e. sounds that are caused by other sources in the room or reflections of the sound that is produced by the buzzer, we apply two conditions on the measurements received, both of which have to be satisfied. First, we apply a 16-amplitude simple moving



(a) Peak-to-peak amplitude



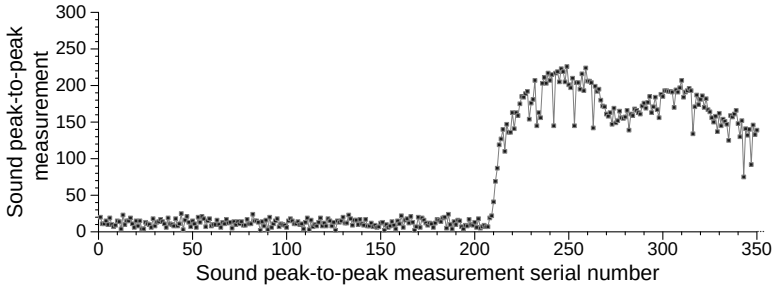
(b) Period (Number of measurements between consecutive peaks)



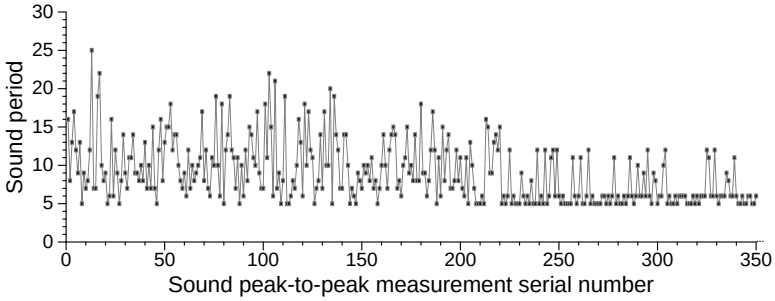
(c) 16-amplitude simple moving average

FIGURE 2.7: Sound measurements at 100cm from sounder.

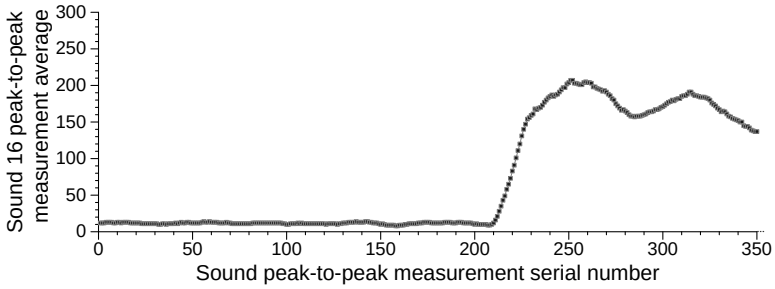
average filter. The average produced by the filter has to surpass a predefined threshold THRESH-A so that we can safely deduce that the sound received is of a certain duration and therefore a potential buzzer sound. The value of THRESH-A may be quite low, since undesired sounds are normally of small duration and averaging makes them look like noise, whilst in the case of a room where persistent sounds are present, it is difficult to identify



(a) Peak-to-peak amplitude



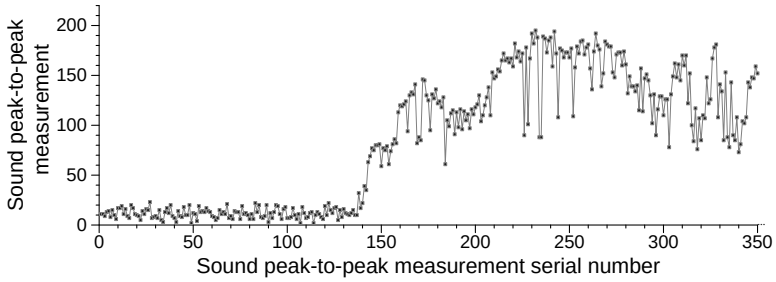
(b) Period (Number of measurements between consecutive peaks)



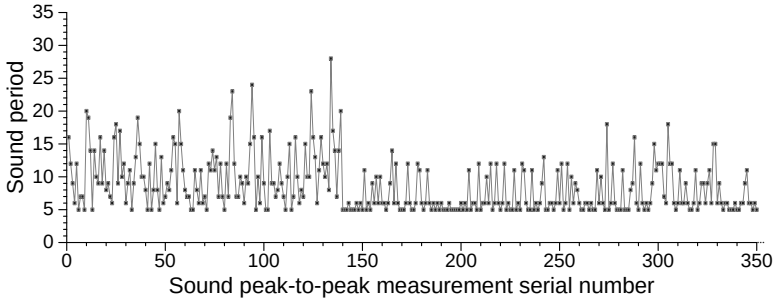
(c) 16-amplitude simple moving average

FIGURE 2.8: Sound measurements at 300cm from sounder.

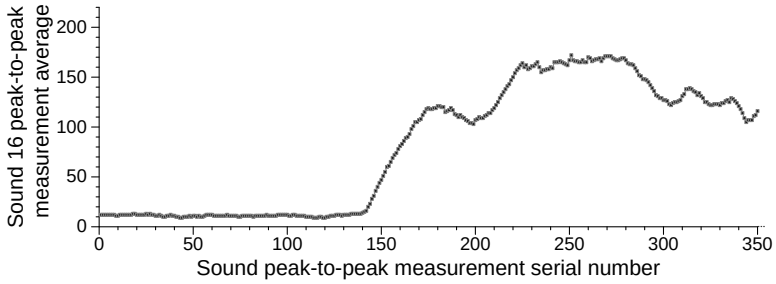
the buzzer sound anyway. In figures 2.7(b), 2.8(b) and 2.9(b) can be seen the 16-amplitude moving average value for a sound pulse of 90ms duration, produced 100cm, 300cm and 500cm away from the listener. Second, we require that the period of the sound wave received, is in accordance to the known period of the sound wave produced by the buzzer.



(a) Peak-to-peak amplitude



(b) Period (Number of measurements between consecutive peaks)



(c) 16-amplitude simple moving average

FIGURE 2.9: Sound measurements at 500cm from sounder.

2.2.4 Copying with the uncertainty at the receiver

Predefining a constant threshold THRESH-B and waiting for the peak-to-peak amplitude of the sound perceived by the listener to surpass it in order to get a timestamp would be a simple thing to implement, however it is not a functional choice. The fact is that the same peak-to-peak amplitude value may correspond to different sound levels, depending on various factors, such

as:

⇒ The sensor that is being used. Different sensors, even if they are of the same type and manufacturer, may produce measurements of different values.

⇒ The mote itself, as variations in reference voltage in ADC converter may introduce inaccuracies in analog to digital conversion.

⇒ Sound levels are definitely not constant with increasing distance and maintaining a constant threshold is not functional. Suppose that sound levels drop with distance, which is the normal thing to suggest, then a constant threshold would identify earlier the sound when the sounder is near as opposed to when the sounder is more far away.

⇒ The nature of sound. Experiments we have conducted demonstrate that sound levels do not decrease with distance, as expected, but there may be anomalies that cause sound levels to increase at some point with increasing distance and then drop abruptly.

That is why the threshold we use must be relative to the peak-to-peak amplitude of the sound pulse, as perceived by the listener. Given that we already maintain the 16 most recent peak-to-peak values and noticing that the wave front has a duration of no more than eight peak-to-peak measurements, it is convenient to use the increase between the most recent measurement and the oldest one of the eight last measurements as a metric. This metric takes its maximum value INC_{MAX} at the beginning of the sound pulse. We use INC_{MAX} to define threshold THRESH-B and to be more exact we set the threshold equal to half INC_{MAX}. As soon as THRESH-B is surpassed by the sound peak-to-peak amplitude a timestamp has to be captured. Figure 2.10 depicts the calculation of the threshold in the case of sound pulse produced at 300cm distance. The reasons that we made such a

choice for the threshold are quite obvious, having in mind the shape figure 2.10. The maximum 8-point increase is essentially proportionate to the magnitude of the received pulse, while being calculated at the the forefront of the pulse makes it a more accurate metric than maximum peak-to-peak values that may induce errors due to sound reflections. Moreover, we chose a fraction of the maximum increase as a threshold since we need the threshold to refer to the part of the graph where the peak-to-peak values are more sparse so that the chance of choosing the right one be greater.

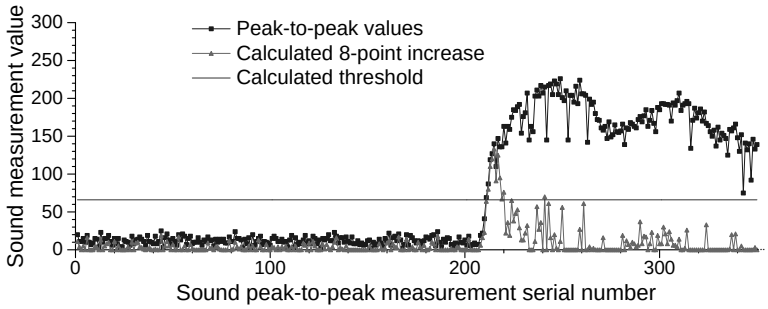


FIGURE 2.10: Sound measurements at 300cm away from sounder with threshold THRESH-B calculated.

2.2.5 Putting it all together

According to our scheme, the sounder produces a pulse of 90ms duration. At the same time it captures its own synchronized timestamp at the moment of raising the pin that will produce the pulse.

At the listener, it is first important to figure out if there exists a sound pulse and then capture the desired timestamp. The listener, having the RF interrupts disabled so that no ADC measurements are lost due to RF interrupt service, is actively waiting for the sound by constantly reading microphone measurements. Every time that a new INC_{MAX} value is located, the 32 most recent peak-to-peak measurements and their corresponding timestamps are stored in a separate buffer. Should the captured INC_{MAX} cor-

respond to the beginning of the sound pulse, then with high probability the peak-to-peak measurement that surpassed threshold THRESH-B can be located among the 32 most recent peak-to-peak measurements.

The listener is certain to have located a sound pulse when the 16-point average value is above threshold THRESH-A over a certain period of time, which we have defined to be equal to 50 peak-to-peak measurements, and the period of the sound perceived is in accordance to the buzzer's frequency. Should this be the case, then the moment that the sound reached the listener is defined to be the moment that THRESH-B was surpassed right before INC_{MAX} was calculated. The listeners goes through a backward procedure calculating the time elapsed since the moment that sound peak-to-peak measurements became greater than the threshold.

After obtaining the sounder's and the listener's timestamps, using an equation like Equation 2.3 should work well in translating time into distance. Having in mind though that speed of sound changes with humidity and temperature, we decided it is be a better choice to fingerprint sound ToF to distance for the certain conditions prevailing in our lab. Results show that ToF increases linearly to distance and given any sound ToF measurement T_F, the distance can be obtained making use of Equation 2.5 where distance DIST is expressed in cm and $T_L - T_S$ is expressed in units of 52us which is the duration of an ADC measurement conversion and thus the maximum achievable granularity. According to this equation sound velocity can be calculated to be equal to 374.6 m/s, however this value is bound to contain a slight error as sensors were synchronized with each other but not with an external timer. One might think that changing the synchronization master mote might yield additional error in distance calculation. Nevertheless this is hardly the case, as we have already proved in 2.1.5 that adjusted synchronous timers have a drift of only microseconds per second. The constant value that exists in Equation 2.5 and is to be subtracted in order to calculate distance is

due to the fact that at the listener, the timestamp is not actually captured at the very beginning of the sound, but at some time later, when sound peak-to-peak amplitude has surpassed THRESH-B. Measurements have shown that Equation 2.5 is consistent for all sensor motes and buzzers and was just needed to be made once and for all. Moreover slight temperature or humidity changes like the ones expected to happen in an indoor-environment do not seem to have a noticeable effect.

$$DIST = 1.948 * (T_L - T_S) - 61.068 \tag{2.5}$$

2.3 Implementation issues

Implementing our scheme was not a simple thing to do, given that sensor motes are resource limited devices. There is no spare memory and most importantly there is no spare energy to waste. And this was our main problem. In specific, while building our system we had to encounter the following challenges:

i) It was not an easy task finding a proper buzzer for the motes. The buzzer has to be able to operate efficiently in supply voltage of 2.5Volt, that can be provided by the sensor mote battery. Furthermore, there exists a limitation to the current it may draw. The INT0 pin can provide slightly more than 10mA, which is not enough for the sounder to function appropriately. It is important that the sounder is powered adequately because if it is not, then at the transient point of start-up which is crucial because all our measurements are based on what we regard as the start of sound, it may behave unexpectedly and lead to erroneous measurements. In order to be sure that the sounder will function flawlessly, we have made use of a transistor so that the

buzzer draws current from the VCC pin which can provide current up to 100mA, every time INT0 is activated.

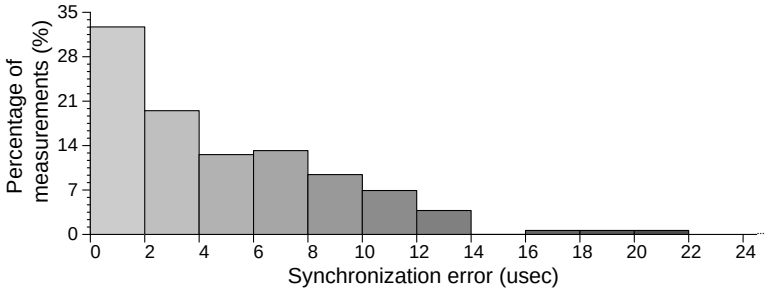
ii) As is already known, the greatest weakness of sensors is the fact that their battery life is quite limited. Having ADC on for reading sound measurements and most importantly activating the buzzer for sound production is rather power consuming for a sensor mote. Energy issues are a challenge to be addressed by future researchers anyway. We have not performed any extensive experiments focusing on how battery life is affected. However, while developing our system we made extensive use of programming boards that provide the motes with current and voltage corresponding to a fully charged battery.

Chapter 3

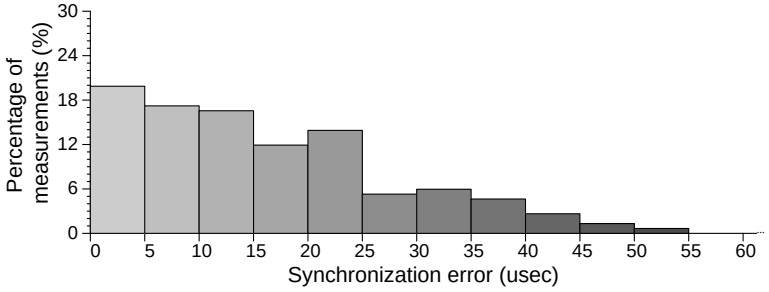
Experimental Evaluation

3.1 Synchronization protocol evaluation

In order to evaluate the efficiency of our synchronization algorithm, we performed a series of experiments on the synchronized motes. For this purpose, we made use of two sensor motes; mote A and mote B. Mote A is the one that periodically, with a period of T seconds, broadcasts the local time timestamps which mote B uses in order to achieve synchronization. Moreover, we had both motes' int0 pin connected to a common switch. This way we were able to periodically, with a period t, produce interrupts to both motes simultaneously and receive their synchronized timestamps referring to the moment that the interrupt was perceived. The absolute value that results by subtracting mote B's timestamps from mote A's timestamps is the synchronization error we are looking for. We performed the experiment twice, once for T=31s and t=19s and once for T=100s and t=35s. Both experiments had a duration of 90 minutes. The results are depicted in image 3.1(a) and image 3.1(b). In the first case the average error is 5us and the maximum error is 20us, whilst in the second case the average error is 16us and the maximum error is 75us.



(a) Synchronization error - 31s resynchronization period



(b) Synchronization error - 100s resynchronization period

FIGURE 3.1: Synchronization error.

3.2 Localization protocol evaluation

We go on into evaluating our localization scheme. For that purpose we have performed a series of experiments that will help us realize the potential and limits of our implementation.

3.2.1 One-dimensional distance

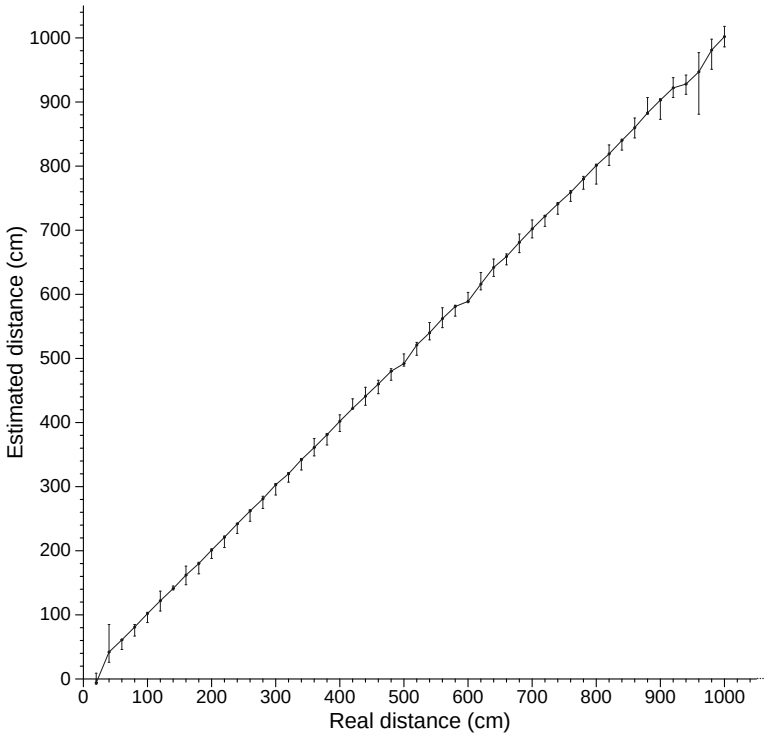
For the purpose of this experiment, we have implemented our localization protocol on two motes. Our goal was to realize the scope of our localization protocol when the sensor motes are in line of sight in a low noise room. We also used a third mote, acting as a gateway mote in order to collect the measurements in a computer and as synchronization master at the same time. Relocalization was taking place every 10s. Beginning from real distance of 20cm between the nodes and increasing it by 20cm every time 20 sound

ToF measurements were obtained, we reached up to the distance of 1000cm. Then we applied the linear equation (Equation 2.5) given in 2.2 in order to estimate distance. In Figure 3.2(a) can be seen the median and 95th percentile of the distance measurements compared to real distance, while in Figure 3.2(b) is depicted the median error in distance calculation with regard to real distance.

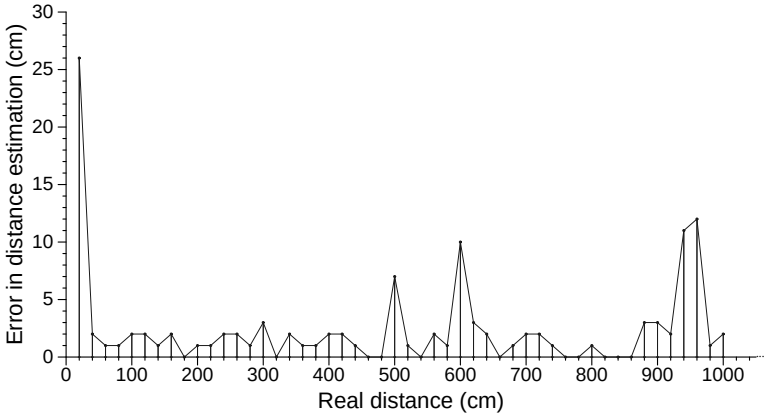
As can be seen in Figure 3.2(b) the error in distance estimation is limited, with an average value of 3cm. It is worth noticing that in most cases estimation error falls within the margin of measurement error, given that the mote size is 2.5cm and its placement was made manually. Variation in distance estimation is also low, and that is mainly because our implementation does not allow for it. This does not mean however that long distance measurements are robust, since sound magnitude in long distances is comparable to noise levels. Apart from long distances, it can be seen that there is significant error in very small distances. This is because of dynamic threshold THRESH-B calculation. In small distances threshold THRESH-B cannot be properly calculated as sound measurements tend to reach the top and bottom values that the ADC can provide, thus INC_{MAX} gets a maximum value which is definitely not representative of the sound magnitude. That is why in small distances, mapping sound ToF to distance cannot be done with a linear mapping function. We have addressed that problem by using a special binomial function on small sound ToF measurements to produce more accurate results in small distance estimation.

3.2.2 Two-dimensional distance

With our next experiment we want to use our localization protocol in order to infer if a simple two dimensional positioning scheme is possible and to demonstrate that distance measurements are neither microphone or sensor dependent. We placed eleven motes in arbitrary positions on the floor on a $5 \times 5m^2$ square room, while two of them were equipped with a buzzer, i.e.



(a) Estimated distance vs real distance



(b) Error in estimated distance

FIGURE 3.2: Localization in one dimension.

node0 and node1. In Figure 3.3 is depicted with squares the exact placement of motes in the room. We programmed node0 and node1 to produce a sound

pulse in turn once every 20s for a total period of 10min. Every time a sounder produces a sound all other motes timestamp the sound arrival and forward their timestamps to the gateway mote. Based on the measurements received in each round, we calculate each mote's distance from the each sounder respectively. The dots in Figure 3.3 depict the placement of motes as was estimated using the median distance from nodeA and nodeB calculated for each sensor. The median localization error is 8cm whilst the maximum localization error is 36cm and corresponds to the error in localizing node 5.

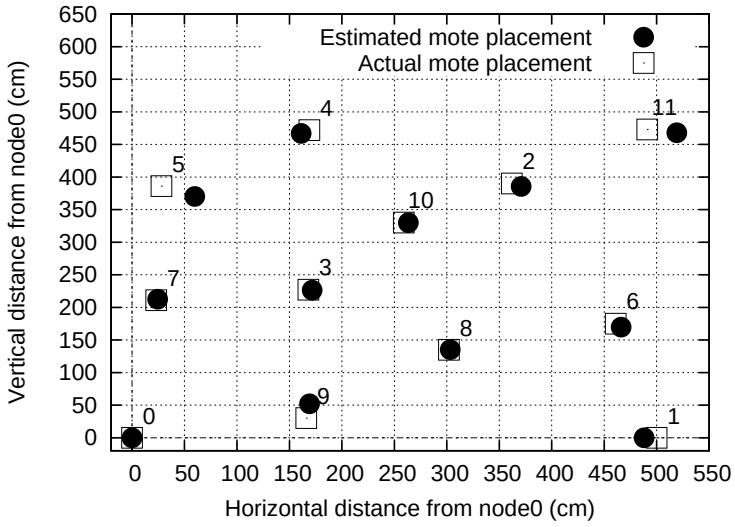


FIGURE 3.3: Localization in two dimensions.

3.2.3 Moving reference point

In case we need to measure distances greater than 5m, the solution is having more sensors produce sound and therefore act as reference points. This way distances can be aggregated. In order to demonstrate that, we have performed the following experiment: We have placed three motes (node0, node1 and node2) producing sound in 5m distance from each other. Among those motes we have placed more motes, that do not have to be equipped

with buzzers. We had the motes that are equipped with buzzers successively produce sound pulses and calculated distances in the sensor network. We have let the localization protocol run for 5 minutes powered up exclusively by batteries. In Figure 3.4 are given in horizontal line 1 the actual placement of motes, while in line 2 is given the estimated placement of motes, having kept for each mote only the distance from the closest reference point. What we should get to realize out of this experiment is that information given by more than one reference points can be used, with some increase in error, to localize motes in long distance, since doing that with one hop acoustic ranging would be impossible.

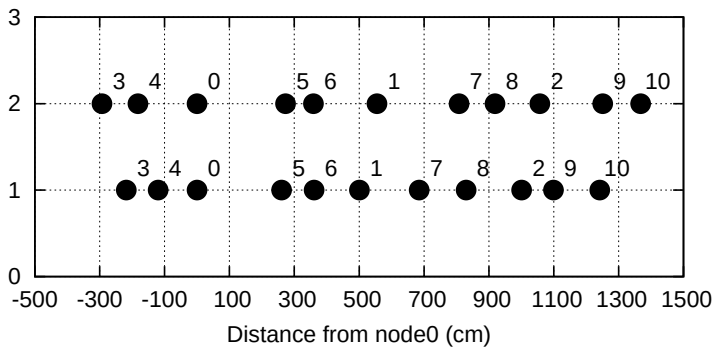


FIGURE 3.4: Moving reference point.

3.2.4 Tracking motion

Finally, we examine how our scheme can be used to track motion in indoor environments. We denote sensors 0 and 1 as reference points and place them at the corners of a $5 \times 5m^2$ square room. We then tie a mote a thread and move it at a speed of 1cm/s on a straight track parallel to the line defined

by the two emitters and 3.05m away from it. Figure 3.5 shows how the reference sensors perceive this motion using the localization scheme. It can be seen that the estimated track diverges on average 8cm and at most 11cm from the real track.

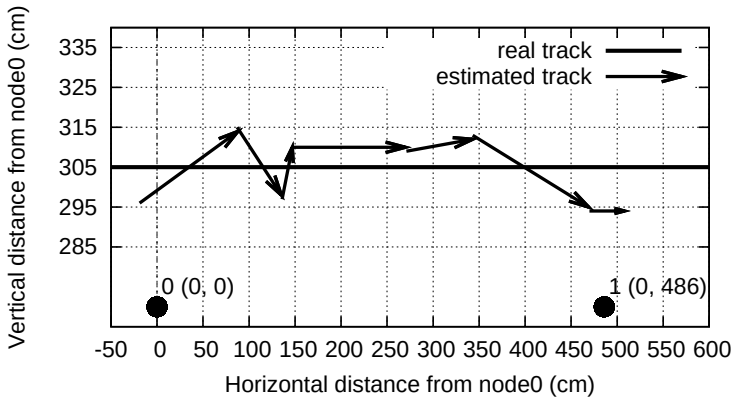


FIGURE 3.5: Tracking motion.

Chapter 4

Limitations and Discussion

In this section we discuss various aspects of our protocol, that we consider important.

4.1 Discussion on synchronization

4.1.1 Scaling with the number of motes

There should be no problem for the protocol regardless of the number of motes comprising the network. The only part of the protocol that could pose a bottleneck is having the motes estimate their transmission delay. However, this procedure entails only one round message exchange. Therefore, if there are collisions, there are back-off mechanisms to ensure that there will be no essential problem. Apart from that, the fact that synchronized timestamps are flooded into the network, makes our protocol extremely scalable with the number of nodes, since the number of synchronized messages remains constant regardless of the mote population.

4.1.2 Scaling with distance

In our protocol, there was no imperative need for multi-hop synchronization, since RF transmission range is many times greater than the maximum audible sound localization range. Nevertheless, the idea about how the scheme

should be adapted to include multi-hop synhronization already exists. In that case, similarly to the current protocol, we can have a primary master node periodically broadcasting synchronized timestamps every T_{active} seconds. Every node that gets synchronized begins broadcasting its own synchronized timestamps every $T_{passive}$ seconds, with $T_{passive}$ being an order of magnitude greater than T_{active}. Unsynchronized nodes, upon receiving their first synchronization message, they address to the sender of the message in order to measure the transmission delay to that mote and thus be able to make use of the synchronization messages it broadcasts in order to get themselves synchronized as well. Synchronized motes that become secondary masters this way, turn their broadcasting period equal to T_{active} seconds, i.e. equal to the primary master's period.

4.2 Discussion on localization

4.2.1 Scaling with the number of motes

There appears to be an issue in localization concerning the number of motes. The number of localization messages produced is in direct dependence on the number of motes. The greater the number of motes, the greater the interval between relocalization phases has to be, so as to guarantee that there will be no packet collisions while forwarding the measurements to the gateway mote. That entails that relocalization takes place less often and node movements are more difficult to be traced in time.

4.2.2 Scaling with distance

As already discussed in 3.2, in even in distances greater than 1000cm localization is possible to produce distance measurements, and even though they are not accurate, they approximate the real distance. The range of localization is not constant for all environments, rather depends on other parameters as well, such as existence of obstacles or noise in room. In great distances we expect the motes not to produce any measurements at all. That is what

they would definitely do in case of lack of noise. However, since noise is rarely absent, there might be cases when measurements are produced when motes mistakenly sense a sound similar to the buzzer sound. In that case, it is the work of the location-graph building program, which is running in the remote computer, to filter out those inconsistent measurements.

4.2.3 3D localization and line of sight

The way that existing obstacles between sensors influence the measurements is not yet examined thoroughly. We should perform more experiments on that. Quite certainly though, the type of obstacle is a major factor in the scale to which the measurements will be affected. Examining how sound is perceived when the sensors are not in line of sight is the most important step towards 3D localization.

4.2.4 Noise

Noise is always a negative factor. Besides, the frequency range of the microphone is wide and there are many sound sources that can produce noise. In the best case, when loud noise is present along with our sound pulse, the sound pulse will be discarded since it will not fulfill the periodicity requirements. In the worst case, a timestamp will be produced, and the remote program will have to filter it out if it is inconsistent.

4.2.5 Monitoring moving motes

Localization of moving nodes can be done with high accuracy. Nevertheless, as stated in 4.2.1 the greater the number of the moving nodes the greater the interval between successive relocalization phases. If nodes move at a high speed, there might be a possibility that the localization system will not be quick enough to capture the nodes' movements.

Chapter 5

Related Work

5.1 Previous work on synchronization

Time synchronization is an important feature of wireless sensor networks and many protocols have been proposed so far. It is important to maintain synchronized clocks when collecting data, so that they can be merged and processed appropriately. Other applications like reliable communication or localization require synchronized timestamping in order to work properly. Moreover, synchronized clocks may be used for energy saving purposes, so that motes enter power saving mode and then return to normal operation at a scheduled time. Most synchronization techniques proposed so far vary both in terms of achievable accuracy and resource consumption.

NTP [20] is the most important network synchronization protocol so far and can provide synchronization accuracy of some milliseconds. In NTP every node synchronizes itself with a master, through a two way message exchange procedure. NTP, however, is not an adequate protocol for sensor networks as it is meant to function in cases where the communication path is not symmetric. It applies well for example in wired computer networks where exist many stochastic factors that affect one way transmission delay. This is not the case in wireless sensor networks, where synchronization takes

place between neighboring nodes. Nevertheless, the basic principle of NTP, which is the way to estimate transmission delay using a two way timestamp exchange procedure is a principle that all protocols adopt, either explicitly by incorporating it into the protocol, or implicitly, in the case where the protocol is proposed for a specific device type and the calculated transmission delay is hardcoded.

Another important protocol that has been proposed is Reference Broadcasting Synchronization technique [7]. In RBS a node transmits a signal and the rest of the nodes use that signal as a time reference. They record their local time at the moment of receiving the reference and thus they estimate their time offset with each other. The most prominent advantage of this approach is that it eliminates all non deterministic procedures at the transmitters side, even though it does not manage to do the same with receiver side processing delays. The main disadvantage of this approach is that the reference sender does not get synchronized along with the rest of the motes and therefore resynchronization needs to take place with another mote being the reference sender, thus resulting in an increase in wireless network load.

Time synchronization Protocol for Sensor Networks [9] is another protocol according to which a sensor mote acting as root synchronizes the rest of the network nodes that are organized into a spanning tree formulation. This protocol demands that each synchronization act between pairs of nodes comprises of a round message exchange during which both the propagation delay and the relative offset of the two clocks are estimated. The main advantage of TPSN is that timestamping takes place at the MAC-Layer, thus eliminating some of the greatest non deterministic delays in message transmission that should otherwise be accounted for. It's main disadvantage is that it requires that every node in each synchronization phase performs the two message exchange thus resulting to $2 \cdot N$ messages needed for a network of N nodes. Moreover, it does not provide a mechanism to compensate for

the clock drift of nodes.

Flooding Time Synchronization Protocol [18] achieves a higher accuracy than the rest of the protocols due to the fact that it operates at MAC-Layer and eliminating unknown delays by means of calibration through multiple timestamping. FTSP uses flooding in order to diffuse synchronized timestamps into the network, thus reducing greatly the wireless network load caused by synchronization. Average synchronization error in Mica2dot nodes where FTSP is implemented is equal to 4us, while maximum synchronization error is 12us. FTSP makes use of linear regression in order to compensate for the drift in between resynchronization periods. The main shortcoming of this protocol is that it is hardware dependent, i.e. the transmission delay time is not estimated dynamically by the protocol, thus requiring calibration on the hardware actually used in the deployment. Moreover, the fact that all motes after becoming synchronized start beaconing synchronized timestamps contradicts the stringent energy consumption constraints.

Delay Measurement Time Synchronization protocol [25] is quite similar to FTSP with the difference that as timing source is used the 32kHz crystal, providing a lesser precision yet a simpler synchronization scheme, which can be implemented in cases where the need for accuracy in synchronization is not imperative. However, DMTS cannot be used in cases where there is a need for high accuracy so that applications may function correctly. One such example is our localization application, where timestamping the arrival of sound has to be of higher accuracy than the one provided by DMTS.

5.2 Previous work on localization

Localization is also an issue that has concerned many researchers so far. There have been proposed various approaches each of which addresses the requirement for localization from a different point of view. For example there is the well-known GPS for outdoor localization, Cricket [26] and AHLoS [27]

that both measure TDoA between RF and ultrasound signals, RIPS[19] and RADAR [3] that use received RF signal properties in order to infer distance. None of these however uses acoustic sound and ToA measurement over a synchronized wireless sensor network. In fact most previous work is based on devices more powerful than sensors, like PDAs or even sensors connected to desktops.

Girod et al. in [10] claim that the resources of Mica motes are not enough for sound detection implemented in software and have them connected to PDAs. The idea is to have PDAs equipped with motes placed in a room localize themselves and form a coordinate system that can be used later on in order to localize mere motes that will be moving in room and produce sound pulses. PDAs and motes are all synchronized in a global timescale using RBS with each node broadcasting a synchronization packet every 10s. The motes act as sounders and the PDAs as listeners. Acousting ranging is performed by having a mote produce a sound pulse and the PDAs that detect the sound send the corresponding sound time series over 802.11 to the PDA that is connected to the mote that produced the sound. Sound time series are correlated using a sliding correlator and looking for the point of maximum correlation. This way the start of sound can be located in time and the distance be calculated. In experiments performed in a lab room $800 \times 1000 cm^2$ wide, the average error in localization is 11.5cm.

Azimi-Sadjadi et al. in [2] have designed a new sensor board using FPGA chip in order to use it in a similar way like PDAs were used in [10]. They had wireless sensor motes connected to FPGA and formed an enhanced sensor unit. The FPGA is equipped with five acoustic channels and an analog to digital converter. Wireless sensors are incorporated into the new sensor solely to provide RF communication and time synchronization. The protocol used for synchronization is FTSP. Whenever a sound is detected in room, the hardware requires a synchronized timestamp by the connected mote and

then sends this timestamp along with the acoustic time series concerning the sound to a base station. In the base station, acoustic time series regarding the detected sound are cross-correlated and the distance from the sound source is calculated through an algorithm that makes use of TDoA. The error in localization is up to 50cm for distances up to 20m from the boards.

Lopes et al. in [16] also have developed a system using PDAs and acoustic sensors. The PDAs are to produce sound pulses that will be sensed by sensors and thus the position of PDAs in room will be located. The sensors are not wireless though. They are wired and connected to desktops, that maintain a synchronized clock. Synchronization with the PDA is temporarily achieved by sending an RF signal that is to notify each PDA to produce a sound and the sensors to listen for the sound. Then TDoA is measured and translated into distance. In their testbed they used 6 sensors in a $23 \times 9m^2$ room and used the measurements of the 3 sensors that were closer to the sound. They manage median error of localization from 5cm to 74cm depending on the position in room.

In Kalamari [30] synchronization and acoustic sound were used in order to infer distance. However in that case sound was sensed with the use of special tone detector hardware. However, the error in their results reaches up to 30cm. Moreover, calibration is required for the system to function adequately.

Chapter 6

Conclusions and Future Work

In this work, we have suggested two schemes; a scheme on synchronization and another one for localization in sensor motes.

Our synchronization protocol was developed on Mica2dot Berkeley motes using CORMOS, an event driven operating system. Nevertheless it could be adapted to function on any sensor architecture. Our scheme comprises of two phases. Every node at first makes a round message exchange with the node from whom the synchronization messages are to be received. Right after that, synchronization takes place by flooding the sensor network periodically with synchronization messages. This way, we have managed synchronization precision of a median of less than 4us, with 30s resynchronization period. The results are satisfactory compared to other previous work both in terms of precision and communication overhead, since synchronized timestamps are broadcasted by a synchronization master. Moreover, we have introduced an external mechanism for testing synchronization precision, i.e. causing simultaneously interrupts to synchronized motes and then comparing the synchronized timestamps produced.

Our synchronization protocol achieves high synchronization precision

without posing great demand on sensor resources. It is more appropriate for short ranges, when all motes are within range of the synchronizing master and has proved to work well with our localization scheme. Besides, the latter fact suggests two things: On the one hand it guarantees that the synchronization scheme does well in terms of resource consumption and, on the other, it validates the synchronization precision that we have claimed to manage. So far we have not built a robust protocol for multihop environments, yet that is a step to be made soon.

Our localization scheme shows that localization can be made with low-cost, off-the-shelf devices and yet be quite precise. Using the Berkeley Mica2dot motes plus the default microphone that already exits on the sensor platform and cheap simple audible sound buzzers we were able to locate nodes in distance up to 1000cm, depending on surrounding noise. The average error in localization precision is 11cm for distances up to 700cm. However, our approach does not require either calibration or any special infrastructure. Furthermore, our method requires a single sounder and microphone per node, resulting in better energy efficiency compared to methods that require multiple sounders and microphones per node.

Finally, we believe that the main issue remaining for future work is to examine our scheme in more demanding environments, such as in the presence of obstacles between nodes, intense noise in room, temperature and humidity variations and outdoor environments.

Bibliography

[1] G. Anastasi, A. Falchi, A. Passarella, M. Conti, and E. Gregori. Performance measurements of motes sensor networks. In *Proc. of the 7th ACM intern. symposium on Modeling, analysis and simulation of wireless and mobile systems (MSWiM'04)*, pages 174–181, New York, NY, USA, 2004.

[2] M. R. Azimi-Sadjadi, G. Kiss, B. Fehér, S. Srinivasan, and A. Lédeczi. Acoustic source localization with high performance sensor nodes. In *Proc. of the SPIE'07 Defense and Security Symposium*, Orlando, FL, April 2007.

[3] P. Bahl and V. Padmanabhan. Radar: an in-building rf-based user location and tracking system. *Proc. of the 19th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2000)*, 2:775–784 vol.2, 2000.

[4] H. Bai, M. Atiquzzaman, and D. Lilja. Wireless sensor network for aircraft health monitoring. *Proc. First Intern. Conference on Broadband Networks (BroadNets 2004)*, 0:748–750, Oct. 2004.

[5] E. Biagioni and K. Bridges. The Application of Remote Sensor Technology to Assist The Recovery of Rare And Endangered Species. *Intern. Journal of High Performance Computing Applications*, 16(3):315–324, 2002.

[6] D. M. Doolin and N. Sitar. Wireless sensors for wildfire monitoring. In M. Tomizuka, editor, *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 5765 of *Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference*, pages 477–484, May 2005.

[7] J. Elson, L. Girod, and D. Estrin. Fine-grained network time synchronization using reference broadcasts. *SIGOPS Oper. Syst. Rev.*, 36(SI):147–163, 2002.

[8] J. Elson and K. Römer. Wireless sensor networks: a new regime for time synchronization. *SIGCOMM Comput. Commun. Rev.*, 33(1):149–154, 2003.

[9] S. Ganeriwal, R. Kumar, and M. B. Srivastava. Timing-sync protocol for sensor networks. In *Proc. of the 1st intern. conference on embedded networked sensor systems (SenSys'03)*, pages 138–149, New York, NY, USA, 2003.

[10] L. Girod, V. Bychkovskiy, J. Elson, and D. Estrin. Locating tiny sensors in time and space: a case study. *Proc. of the 2002 IEEE Intern. Conference on Computer Design: VLSI in Computers and Processors*, pages 214–219, Sept. 2002.

[11] A. Hande, T. Polk, W. Walker, and D. Bhatia. Self-powered wireless sensor networks for remote patient monitoring in hospitals. *Sensors*, 6(9):1102–1117, 2006.

[12] M. Klann, T. Riedel, H. Gellersen, C. Fischer, M. Oppenheim, P. Lukowicz, G. Pirkl, K. Kunze, M. Beuster, M. Beigl, O. Visser, and M. Gerling. Lifenet: an ad-hoc sensor network and wearable system to provide firefighters with navigation support. In *Adjunct Proc. Ubicomp 2007*, pages 124–127, sep 2007.

[13] V. A. Kottapalli, A. S. Kiremidjian, J. P. Lynch, E. Carryer, T. W. Kenny, K. H. Law, and Y. Lei. Two-tiered wireless sensor network architecture for structural health monitoring. *Smart Structures and Materials 2003: Smart Systems and Nondestructive Evaluation for Civil Infrastructures*, 5057(1):8–19, 2003.

[14] L. Krishnamurthy, R. Adler, P. Buonadonna, J. Chhabra, M. Flanigan, N. Kushalnagar, L. Nachman, and M. Yarvis. Design and deployment of industrial sensor networks: experiences from a semiconductor plant and the north sea. In *SenSys '05: Proceedings of the 3rd intern. conference on Embedded networked sensor systems*, pages 64–75, New York, NY, USA, 2005.

[15] Q. Li, M. D. Rosa, and D. Rus. Distributed algorithms for guiding navigation across a sensor network. In *MobiCom '03: Proc. of the 9th annual intern. conference on Mobile computing and networking*, pages 313–325, New York, NY, USA, 2003.

[16] C. V. Lopes, A. Haghighat, A. Mandal, T. Givargis, and P. Baldi. Localization of off-the-shelf mobile devices using audible sound: architectures, protocols and performance assessment. *SIGMOBILE Mob. Comput. Commun. Rev.*, 10(2):38–50, 2006.

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

[18] M. Maróti, B. Kusy, G. Simon, and Ákos Lédeczi. The flooding time synchronization protocol. In *Proc. of the 2nd intern. conference on embedded networked sensor systems (SenSys'04)*, pages 39–49, New York, NY, USA, 2004.

[19] M. Maróti, P. Völgyesi, S. Dóra, B. Kusý, A. Nádas, Ákos Lédeczi, G. Balogh, and K. Molnár. Radio interferometric geolocation. In *Proc. of the 3rd intern. conference on Embedded networked sensor systems (SenSys'05)*, pages 1–12, New York, NY, USA, 2005.

[20] D. L. Mills. Internet time synchronization: The network time protocol. *IEEE Transactions on Communications*, 39:1482–1493, October 1991.

[21] T.-K. Moon and T.-Y. Kuc. An integrated intelligent control architecture for mobile robot navigation within sensor network environment. *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2004)*, 1:565–570 vol.1, Sept.-2 Oct. 2004.

[22] N. Noury, G. Virone, and T. Creuzet. The health integrated smart home information system (his2): rules based system for the localization of a human. *Microtechnologies in Medicine and Biology 2nd Annual Intern. IEEE-EMB Special Topic Conference on*, pages 318–321, May 2002.

[23] M.-S. Pan, C.-H. Tsai, and Y.-C. Tseng. Emergency guiding and monitoring applications in indoor 3d environments by wireless sensor networks. *Int. J. Sen. Netw.*, 1(1/2):2–10, 2006.

[24] H. Park, J. Burke, and M. B. Srivastava. Design and implementation of a wireless sensor network for intelligent light control. In *IPSN '07: Proc. of the 6th intern. conference on Information processing in sensor networks*, pages 370–379, New York, NY, USA, 2007.

[25] S. Ping. Delay measurement time synchronization for wireless sensor networks. *tech. rep., Intel Research*, 2003.

[26] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan. The Cricket Location-Support System. In *6th ACM MOBICOM*, pages 32–43, Boston, MA, August 2000.

[27] A. Savvides, C.-C. Han, and M. B. Strivastava. Dynamic fine-grained localization in ad-hoc networks of sensors. In *Proc. of the 7th annual intern. conference on Mobile computing and networking (MobiCom'01)*, pages 166–179, New York, NY, USA, 2001.

[28] I. Stoianov, L. Nachman, S. Madden, and T. Tokmouline. Pipeneta wireless sensor network for pipeline monitoring. In *IPSN '07: Proceedings of the 6th intern. conference on Information processing in sensor networks*, pages 264–273, New York, NY, USA, 2007.

[29] H. Wang, J. Elson, L. Girod, D. Estrin, and K. Yao. Target classification and localization in habitat monitoring. *Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03). 2003 IEEE Intern. Conference on*, 4:IV–844–7 vol.4, April 2003.

[30] K. Whitehouse. "the design of calamari: an ad-hoc localization system for sensor networks.". *Master's Thesis, University of California at Berkeley*, 2002.

[31] J. M. R. Xuehai Bian, Gregory D. Abowd. Using sound source localization in a home environment. *Lecture notes in computer science*, 3468:19–36, 2005.

[32] J. Yannakopoulos and A. Bilas. Cormos: a communication-oriented runtime system for sensor networks. *Proc. of the Second European Workshop on Wireless Sensor Networks*, pages 342–353, 31 Jan.-2 Feb. 2005.