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

Evangelos Mangas[†] and Angelos Bilas[†]
Institute of Computer Science (ICS)
Foundation for Research and Technology - Hellas
Heraklion, GR-70013, Greece
Email: {emangas,bilas}@ics.forth.gr

Abstract

Sensor localization in wireless sensor networks is an important component of many applications. Previous work has demonstrated how localization can be achieved using various methods. In this paper we focus on achieving fine-grained localization that does not require external infrastructure, specialized hardware support, or excessive sensor resources. We use a real sensor network and provide measurements on the actual system. We adopt a localization approach that relies on acoustic sounds and clock synchronization. The contribution of our work is achieving consistent sound pulse detection at each sensor and precise range estimation using a high-precision clock synchronization implementation. We first describe our technique and then we evaluate our approach using a real setup. Our results show that our approach achieves an average clock synchronization accuracy of 5μs. We verify this accuracy using an external global clock via an interrupt mechanism. Our sound detection technique is able to consistently identify sound pulses up to 10m distances in indoor environments. Combining the two techniques, we find that our localization method results in accurate range estimation with an average error of 11cm in distances up to 7m and in consistent range estimation up to 10m in various indoor environments.

1. Introduction

Over the last few years, wireless sensor networks have been used in various application domains: indoor or outdoor monitoring, health applications, structural monitoring of buildings, smart house applications, and robot navigation or even emergency navigation, when a target needs to be directed through a region. Most of these applications require some sort of location awareness, e.g. for recording coordinates with sampled information or to detect proximity. In most cases, systems used in these applications are statically built, that is location information is set once for each sensor at system configuration time. This however, is not only cumbersome, but limits possible applications as well only to domains where static configuration is possible and there is no motion involved, e.g. of objects or humans.

A more practical approach is to provide sensors with the ability to calculate relative positions using some range estimation mechanism and reference points. A number of such techniques have been proposed that use various methods for estimating distances based on the speed of either acoustic sound [1]–[4] or ultrasound [5]–[7], using GPS receivers, infrared [8] or even RSSI [9], [10]. These approaches differ in the resources and support they require both in terms of the surrounding environment and the sensors themselves, the range they are able to cover (typically short or long range methods) and the accuracy they provide.

Our approach uses acoustic ranging that measures distance by calculating time of flight for sound pulses. This approach is less susceptible to interference than other types of pulses (e.g. RF), audible sound is omni-directional, and ranging can be implemented with low-cost hardware that can be embedded in the nodes without posing any extra demand for power.

Previous work using acoustic ranging techniques has examined how localization can be achieved through accurate sound detection. The most prominent approaches apply signal processing methods in software such as correlating the sound time series obtained using a sliding correlator and looking for the point of maximum correlation [1], [2] or embedding tone detectors in order to detect sound by specialized hardware [3]. Previous work aims at lower accuracy, short-range localization, evaluates proposed techniques in simulated environments, or requires specialized support. Instead, in our work we achieve fine-grained localization, at the level of tens of centimeters, primarily for indoor environment applications, without requiring any additional infrastructure in the environment, nor any sensor hardware support or excessive sensor resources.

Acoustic sound localization techniques can be categorized broadly in differential time of arrival (DToA) [4] or simply time of arrival (ToA) techniques [1]–[3]. In DToA tech-

. [†]Also, with the Department of Computer Science, University of Crete, P.O. Box 2208, Heraklion, GR-71409, Greece.

niques, a node emits simultaneously two different types of signal at once, such as an RF and ultrasound or audible sound signal. The RF signal, whose speed of transmission is much faster triggers the start of counting at the listener till the sound signal arrives. This allows the listener to calculate distance based on the speed of the second signal and the time elapsed from the arrival of the first signal. DToA has two main disadvantages: First, using a local timer in calculations may cause erroneous measurements from node to node, due to drift between timers. Second, switching to listening for the second sound signal right after listening for the RF signal can be a high-overhead task, especially when nodes are close to each other and switching overhead can impact the accuracy of measurements. ToA techniques on the other hand, maintain synchronized clocks independently and use these clocks to measure the time of flight for a second, e.g. acoustic sound signal, based on some coordination protocol. Nodes then use time of flight measurements and sound speed to estimate point to point distance.

In our work we use a ToA-based method. Our approach relies on two techniques: employment of a novel method for consistent sound pulse detection along with high-precision clock synchronization via RF communication. Furthermore, we implement and evaluate our approach on a real sensor network that consists of Mica2dot sensors in various configurations. Our clock synchronization technique uses only RF communication to exchange clock values. The strong points of our approach is the accuracy in estimating communication overheads during message exchange and the design of a protocol that results in high precision, incurs low overhead, scales to large numbers of nodes and accounts for clock drift.

Our clock synchronization technique is designed to operate at the MAC-layer. It calculates transmission delays during message exchange, takes into account interrupt and RF communication protocol overheads, and does not require calibration for different types of sensors (and antenna chipsets). For scalability purposes, we use message broadcasts to allow multiple sensors to synchronize with a single master and we avoid any point to point communication between slave and master that would impose a high overhead on the master. Our results on a real system show that our approach achieves clock synchronization with an error of 5μs. To verify the precision of our approach we perform controlled measurements with a global external clock via interrupts.

Our localization technique using acoustic sound relies on consistent sound pulse detection. Each node is equipped with an off-the-shelf, low-cost buzzer that generates sound pulses at an audible frequency of 2.4 kHz and a microphone. Unlike most techniques proposed so far, all the processing is done online by the sensor node. Each sensor serving as a reference point generates a series of sound pulses with a predetermined period. All other sensors detect arrival of these pulses using a low-overhead calculation and filtering. By having the reference point broadcast the corresponding timestamp after every sound pulse, each sensor can estimate the time-of-flight for the sound pulse and calculate point to point distance.

We implement our localization technique, *FLASH*, on the Mica2dot [11] coin-sized motes that use an 8-bit, 3.6MHz AVR processor, communicate via RF at 915MHz, and incorporate sound sensing and generation ability. The runtime system on the sensors is CORMOS [12], an event-based environment that uses the notion of the communication path in task execution. To the best of our knowledge, in such a restricted environment, *FLASH* is the first technique to manage estimation of relative position of sensors within 11cm for single-hop distances of up to 7m and in various setups. Moreover, range estimation behaves consistently for single-hop distances of up to 10m and allows for the use of multiple reference points to cover larger areas in a multi-hop manner.

The rest of this paper is organized as follows. Section 2 presents the design of FLASH and the clock synchronization and sound detection techniques it uses. Section 3 presents our experimental setup and results for indoors environments. Section 4 discusses related work, whereas Section 5 draws our main conclusions.

2. System Design and Implementation

2.1. Clock Synchronization

The clock synchronization scheme we use relies on calculating message delays between nodes, exchanging local clock values, and combining the two to build a single (logical) synchronized clock. Next, we discuss (a) how point-to-point message delays are calculated, (b) how we generate and use timestamps with physical clocks, and (c) how we deal with clock drift.

2.1.1. Message Delays. Previous work [13], [14] has already categorized the main stages during sensor RF communication as (Figure 1): *send time* that includes message creation time above the MAC layer and is non-deterministic and typically in the order of milliseconds, *access time* that includes the time required to get access to the channel and is non-deterministic and typically in the order of milliseconds, *transmission time* that includes the time required for message transmission and depends mostly on RF speed and the communication subsystem employed, *propagation time* that is the time of flight between the sender's and the receiver's antenna, depends almost exclusively on the distance between the antennas, *reception time* that includes the time required for the receiver to receive the message and has similar characteristics to transmission time, and *receive time* that

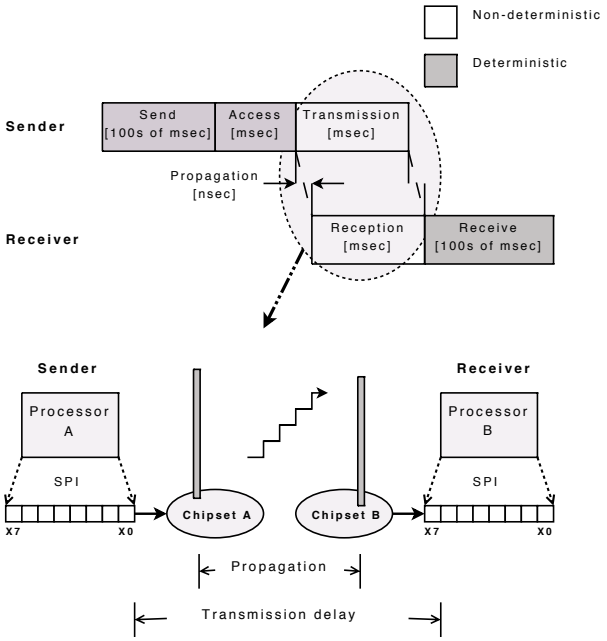


Figure 1. Stages of message delivery.

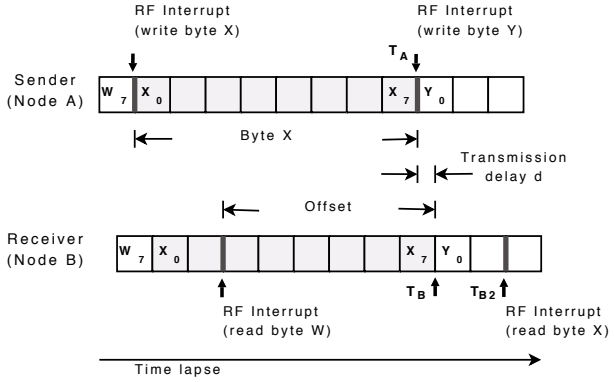


Figure 2. Message timestamping.

is the amount of time needed for the receiver application to be notified of the received message.

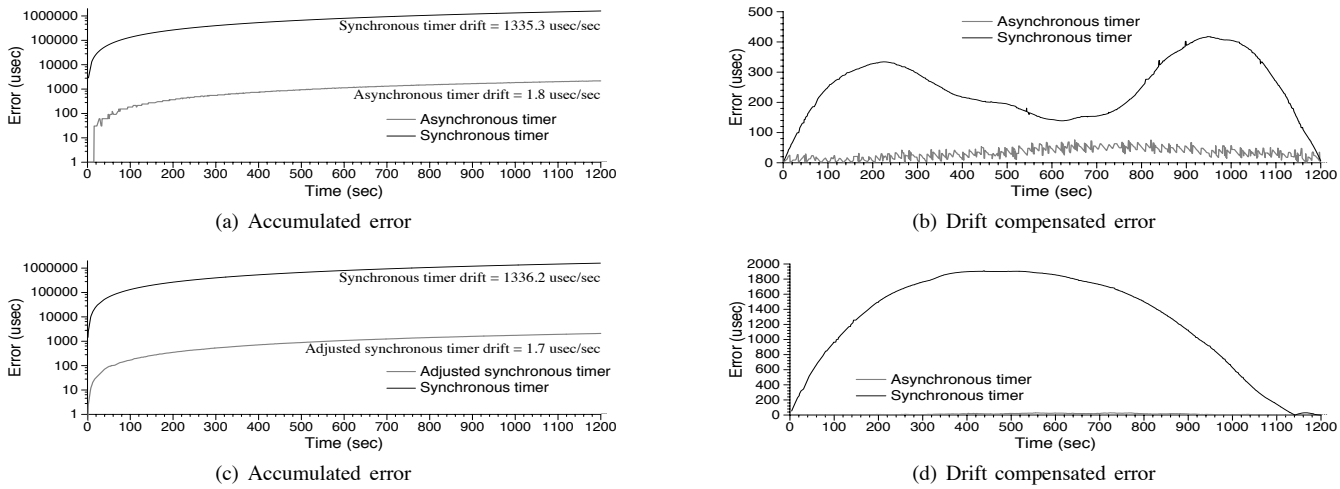
We eliminate the uncertainty of the higher layers for send, access, and receive time by developing a protocol at the MAC-Layer, as close to the transmission phase as possible. This allows us to focus on transmission, propagation and reception delays. Propagation time can be calculated based on the speed of light (3×10^8 m/s). For a distance of 135m, which is the maximum transmission range for Mica2dot motes [11], propagation time is less than 0.5μs. Since we target that are significantly smaller, in the rest of our analysis we ignore propagation time.

Estimating the transmission and reception phase is more involved. For simplicity, let's 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. The steps that take place during transmission are as follows:

(i) The sender writes a byte to the antenna interface (SPI interface). (ii) The byte gets shifted into the antenna at a rate determined by the antenna transmission rate. When the last bit has been shifted into the antenna, an interrupt is raised to the processor. After a time delay equal to the interrupt routine delay the processor will write the next send byte to the interface buffer. (iii) Every bit that enters the antenna needs some time to get modulated into electromagnetic waves. Although modulation time may vary in different antenna chipsets and configurations, it is quite stable for each type of chipset and configuration. (iv) Modulated bits are propagated, demodulated at the receiver's antenna chipset, and shifted into the receiver SPI register. (v) As soon as eight consequent bits have been shifted into the SPI register, an interrupt is raised and the processor collects the received byte and handles it, typically storing it temporarily in a buffer until the whole message is received. (v) Usually, the receiver receives bytes in different alignment than the sender byte alignment. Therefore, the interrupt at the receiver is not raised at the time the actual byte has been received, but rather, when the byte and a few extra offset bits have been shifted into the SPI register. These offset bits either belong to the next byte in the transmitted sequence or are just noise.

For accurate clock synchronization it is essential to measure the exact transmission delay $d = T_B - T_A$ (Figure 2). The transmission delay is the time between the last bit of the transmitted byte has been shifted into the sender's antenna (timestamp T_A) and the time the last bit of the same byte has been shifted into the receiving (SPI interface) register at the receiver (timestamp T_B). Therefore, we need to accurately capture local timestamps T_A and T_B.

At the receiver we generate the local timestamp within the interrupt routine that is executed right after the full byte has been received and then we account for the time that was needed for the extra (8 minus offset) bits that have been received because of the misalignment. The interrupt service delay is in the range of μs and may vary significantly only if another interrupt routine is being executed at the time with the receiving (SPI interface) interrupt. We handle this uncertainty by capturing multiple timestamps for a sequence of transmitted bytes. Thus, we may estimate the amount of time any byte needs to be transmitted at the receiver by using a median filter. Using this self-calibration mechanism before the byte of interest, allows us to compensate for any delay noticed in the execution of the interrupt routine for the byte of interest. A similar mechanism is used on the transmitting side as well. Timestamps T_A, T_B are then used to provide an accurate value for the message delay d, as described next.



(a) Accumulated error

(b) Drift compensated error

(c) Accumulated error

(d) Drift compensated error

Figure 3. Error between synchronous and asynchronous physical (top) and logical (bottom) timers, without and with drift compensation.

2.1.2. Short-term Synchronization. The main idea for transmission delay calculation is similar in almost all clock synchronization protocols: two nodes (A and B) exchange timestamps through a symmetric transaction. The timestamps are captured as described in 2.1.1 so that $T_{1A} + d$ and T_{1B} refer to the same moment in absolute time. The same is true for T_{2B} and $T_{2A} - d$. Thus, assuming clock skew introduced between timers during T_{1A} to T_{2A} period is negligible:

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

Every sensor node A in the network first measures d between itself and a pre-assigned or elected master B. This procedure need occur only once, as long as the same node B is used as master. Then, node B periodically broadcasts synchronization messages that contain synchronized clock timestamps. Upon receiving a synchronization timestamp $T_B, T_B = T_A - d$ and node A can use T_B to compensate for the offset between the two clocks. As already argued in [15] nodes should not set their clock or discipline their frequency, rather than let it run in its natural rate. In accordance to that, node A stores timestamps $T_B, T_A - d$ over a time window and uses linear regression of this sequence to project local time to synchronized time.

2.1.3. Long-term Synchronization. In Mica2dot there are two timer clock sources available. The timer can be clocked either synchronously at 3.6 MHZ (3,686,400 Hz) or asynchronously by an external crystal of 32,768 Hz frequency. We evaluate drift for the two different timer sources, by simultaneously producing a series of external interrupts to pairs of motes. Each mote, at the beginning of the interrupt routine execution produces a timestamp using the synchronous time source and another one using the asynchronous time source and broadcasts them. We collect the timestamps to an external PC system. Suppose that the first timestamps collected were used to synchronize clocks, we calculate the accumulated and the drift compensated error for both timer sources, as given in Figures 3(a) and 3(b). We find that fine-granularity, synchronous timers introduce a drift in the range of milliseconds per second, whereas coarse-granularity asynchronous timers introduce a drift in the range of microseconds per second.

For that reason, we use the asynchronous timer to adjust a virtual synchronous timer. This results in a virtual clock that has the granularity of the synchronous timer and the drift of the asynchronous one. As shown in Figures 3(c) and 3(d) the virtual clock error remains in the scale of microseconds for long time periods, thus allowing for longer resynchronization intervals reducing communication and computation overheads. For long term synchronization every node maintains a set of the four most recently formed pairs of local and remote timestamps, all of which are produced using the virtual clock, and uses linear regression to calculate synchronized time st(T_A) for any given local timestamp T_A.

This scheme allows for synchronizing a single master with all nodes in range of its RF antenna (maximum of 135m for our setup): Resynchronization intervals can be long and only message delays need to be calculated in a pairwise manner between each node and the master. This calculation is very infrequent as it only depends on the type of the sensor and the environment conditions. Our scheme can be extended to multi-hop synchronization, but we do not examine this in this work.

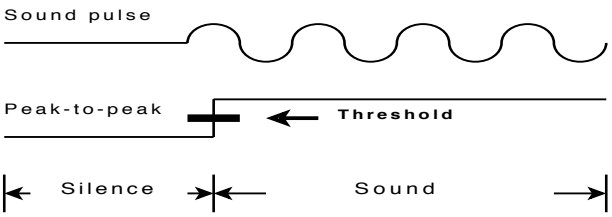


Figure 4. Ideal sound detection.

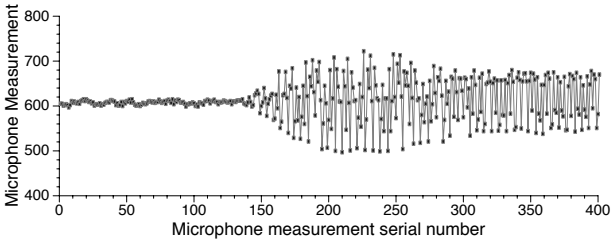


Figure 5. Raw sound measurements.

2.2. Sound-based Localization

We now describe our sound-based localization scheme. For simplicity, let's assume we have a network of two synchronized sensor nodes. The basic idea, as discussed in previous work [1], [2], is to have one node produce a sound pulse while the other listens to the audio frequency spectrum. The node that produces the sound pulse (emitter) generates a timestamp T_S right before the sound is produced. The listening node (listener) generates a timestamp T_L at the moment the timestamp is detected. The two timestamps are generated in a global, synchronized scale and $T_L - T_S$ is the sound time-of-flight (ToF), i.e. the time needed for the sound to travel from the emitter to the listener. Given that the speed of sound in the air is $V_S = 344m/s$, the distance between the two nodes can be calculated as $Dist = V_S \cdot (T_L - T_S)$.

Timestamp T_S can be easily generated and refers to the moment the emitter applies high voltage to the general purpose input/output (GPIO) pin where the buzzer is connected. Generating timestamp T_L in the listener to ensure proper marking of the start of the received sound pulse is more challenging. 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 generate a timestamp at the beginning of the ADC interrupt handler call which corresponds to the first measurement that we accept (detect) as the starting point of the sound pulse. A new ADC measurement is produced every 52μs (208 processor cycles), which means that the maximum feasible accuracy is equal to 1.7cm. In this process, the role of the clock synchronization scheme is to ensure that the synchronization error is at any moment less than 52μs, so that T_L will not mistakenly refer to a wrong ADC interrupt, increasing maximum feasible distance accuracy to more than 1.7cm.

Beyond clock synchronization, we also need to cope with the uncertainties posed by sound creation, propagation, and reception, as discussed next.

2.2.1. Sound Devices. In our work we use the default, low-cost, microphone of 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 17.7KHz. In order to to detect a sound pulse of a certain frequency we have to be able to reconstruct the sound pulse at the receiver. This requires that the sound pulse is of less than 8.85KHz frequency, half the sampling rate frequency. Having the acoustic time series corresponding to the sound pulse available, to locate the beginning of the pulse in time we can apply signal processing techniques, but doing so entails more resources than the ones the sensor can dispose. Otherwise, we may take advantage of the sound pulse's properties, that is amplitude and frequency, and build a dynamic online processing mechanism that locates the first noticeable peak of the sound pulse and avoid excessive storage and processing requirements.

Since Mica2dot sensors do not include a buzzer, we use EMX-7T01SP 1.5V external buzzer devices, at 2.3KHz. The buzzer is controlled by the INT0 GPIO pin of the sensor. Connecting external buzzers to each mote requires a lot of experimentation. The buzzer has to be able to operate efficiently in supply voltage of 2.5Volt that can be provided by the sensor mote. Moreover, there is a limitation in how much current it may draw. The recommended INT0 pin for connecting the buzzer can provide slightly more than 10mA, which is not enough for the buzzer to function properly. Thus, it is important that the mote generating the sound pulse is powered adequately, otherwise, it may not generate reasonably shaped sound pulses, which will result in false detection at the receiver and erroneous measurements. For this reason, we use a transistor to ensure that the buzzer draws current from the Vcc pin which can provide current up to 100mA, when INT0 is activated.

2.2.2. Theoretical Scheme. Ideally a single tone sound-wave that appears in room has the sinusoid waveform given in Figure 4. Figure 5 shows the raw samples for a sound pulse generated at 300cm distance from the listener. Our scheme relies on two main steps: (a) Calculating peak-to-peak values for this signal, which results in a graph with a sharp increase at the start of the sound. (b) Applying a dynamically calculated threshold (THRESH-A), which allows us to identify the start of the sound and attach timestamp T_L to the right (first) peak of the sound-wave.

2.2.3. Scheme Adaptation. Given that the emitter frequency is 2.3KHz, whereas the sampling frequency is 17.7KHz, it takes 7 to 8 ADC conversions to sample a single period of the signal. We identify as sound-wave peaks the measurements whose value is greater than four previous measurements and greater or equal to four subsequent measurements and subtract the lowest value between consecutive peaks. This allows us to calculate the peak-to-peak amplitude. We also use elapsed time between consecutive peaks to calculate the period of the sound pulse. Figure 6 shows the peak-to-peak amplitude and period of the sound as calculated by the measurements taken at the beginning of a sound pulse, for distances of 100cm, 300cm, and 500cm. In all cases the transition from silence to sound is apparent, yet with increasing distance the starting point of the sound pulse becomes less clear.

THRESH-A has to be dynamically calculated, based on the properties of the actual sound pulse. The reason is that the same peak-to-peak amplitude may correspond to different sound levels due to, e.g. difference in sensor components, variations in reference voltage, or just characteristics of sound propagation in the environment.

Noticing that the sound pulse needs no more than eight peak-to-peak measurements to reach its full magnitude, we use as metric the increase between the most recent measurement and the oldest one of the eight previous measurements. As can be seen in Figure 7, the increase takes its maximum value INC_{MAX} at the beginning of the sound pulse. We use INC_{MAX} to define THRESH-A as $INC_{MAX}/2$. Since INC_{MAX} refers to the beginning of the sound pulse, THRESH-A is free of distortion caused by sound reflections. We only use a fraction of INC_{MAX} as a threshold because we need THRESH-A to correspond to the part of the graph where peak-to-peak values are sparser, so that the likelihood of choosing the right one is greater.

2.2.4. Average Filtering. To avoid detecting false sounds in indoors environments, e.g. due to noise by other sources in the room or reflections of the sound pulse, we apply two types of filtering on the measurements: First, we apply a 16-amplitude simple moving average filter. The average produced by the filter has to surpass a predefined threshold (THRESH-B) to ensure that the sound received is of a certain duration and therefore a potential ranging sound pulse. The value of THRESH-B will typically be quite low, since undesired sounds are normally of small duration and averaging makes them look like noise. Naturally, in case of an environment with persistent sounds at the same frequency as the buzzer, it is difficult to identify the ranging pulses. Figure 6 shows the impact of this filter on a sound pulse of 90ms duration, produced at 100cm, 300cm, and 500cm distance. Second, we require that the period of the sound wave received agrees with the pre-defined period used for the ranging sound pulses in the localization protocol.

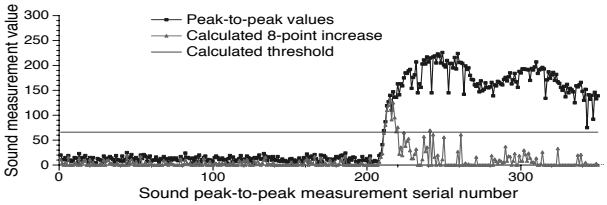


Figure 7. THRESH-A calculation at 300cm.

2.2.5. Implementation. In our scheme, the emitter produces a pulse of 90ms duration and generates a timestamp right before raising the pin that will produce the pulse. The listener detects the sound pulse and then generates a timestamp. The listener disables RF interrupts when listening for ranging pulses to avoid losing ADC measurements when polling the microphone. Whenever a new INC_{MAX} value is sampled, the 32 most recent peak-to-peak measurements and their corresponding timestamps are stored in a buffer. Should the captured INC_{MAX} correspond to the beginning of the sound pulse, then with high likelihood the peak-to-peak measurement that surpassed THRESH-A exists among the 32 most recent peak-to-peak measurements and that is how the listener captures T_L.

The listener starts the look-up process only when (a) it has detected the sound pulse with certainty, that is when the 16-point average value is above THRESH-B for a certain period of time, which we experimentally determine to be 50 peak-to-peak measurements, and (b) the period of the sound pulse agrees with the buzzer's frequency.

After obtaining the emitter's and the listener's timestamps we estimate sensor node distance. To account for sound speed variations due to humidity and temperature, we calibrate once sound ToF vs. distance for the surrounding (experimental) environment: ToF increases linearly to distance and given any sound ToF measurement T_F, the distance can be obtained making use of Equation 2, where distance DIST is expressed in cm and $T_F = T_L - T_S$ in units of 52μs, which is the duration of an ADC conversion and thus, the maximum achievable granularity. Based on this equation sound speed is calculated to 374.0 m/s.

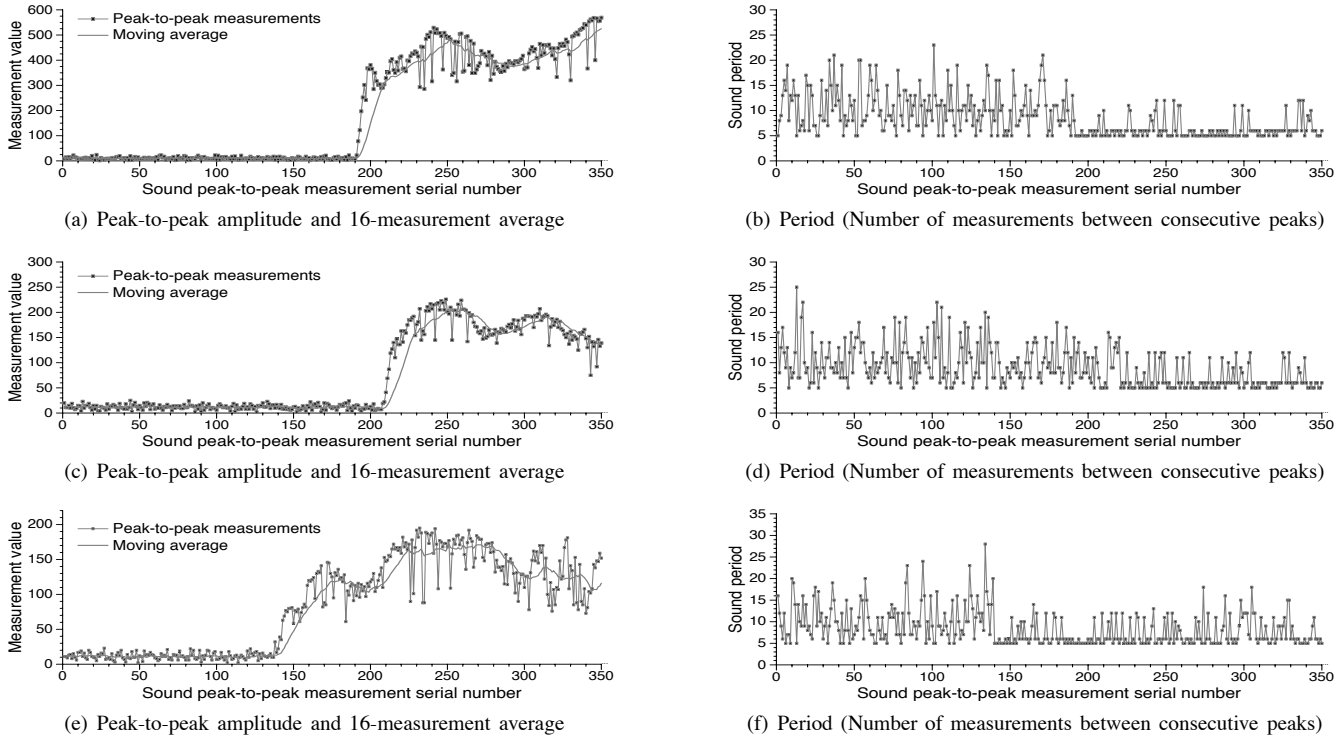
$$DIST = 1.948 * T_F - 61.068 \quad (2)$$

Slight temperature or humidity changes like the ones expected to happen in a specific indoor-environment do not seem to have a noticeable effect. Larger variations will require re-calibration of the system.

3. Experimental Evaluation

3.1. Clock Synchronization

To evaluate our clock synchronization scheme, we perform a series of experiments on the synchronized motes.



(a) Peak-to-peak amplitude and 16-measurement average

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

(c) Peak-to-peak amplitude and 16-measurement average

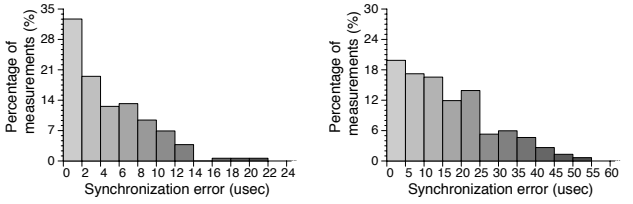
(d) Period (Number of measurements between consecutive peaks)

(e) Peak-to-peak amplitude and 16-measurement average

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

Figure 6. Sound measurements at 100cm (top), 300cm (middle), and 500cm (bottom).

We use an experimental setup that allows us to verify synchronization error. One mote (A) broadcasts its local time periodically (with period T) and another mote (B) uses these values to synchronize its clock. Both motes are connected via their INT0 pin to a common switch. Periodically (at period t) we generate interrupts to both motes simultaneously and receive their synchronized timestamps for the specific (global) interrupt event. The absolute value that results by subtracting the motes' timestamps is the synchronization error. We perform the experiment twice, for T=31s and t=19s, and for T=100s and t=35s. Both experiments have duration of 90 minutes. Figures 8(a) and 8(b) depict our results. In the first case the average error is 5μs and the maximum error is 20μs, whilst in the second case the average error is 16μs and the maximum error is 75μs, resulting in an average error of less than 11μs for both experiments.



(a) Synchronization error - 31sec resynchronization period

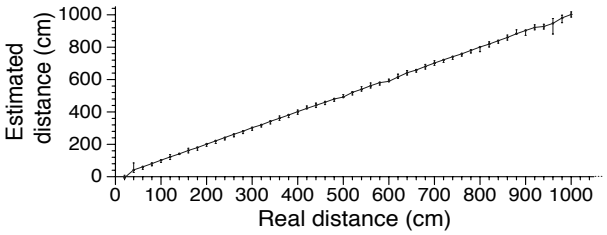
(b) Synchronization error - 100sec resynchronization period

Figure 8. Clock synchronization results.

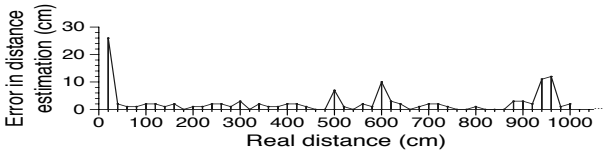
3.2. Sound Localization

3.2.1. Effective Ranging Distance. First, we use a 1-D setup to examine the maximum distance at which our approach is effective, using a pair of sensors. We use a third mote as a gateway for forwarding distance measurements to a PC as well as a clock synchronization master. Each distance estimation uses 20 sound ToF measurements. We increase the distance between the sensors from 20cm to 1000cm with increments of 20cm. Figure 9(a) shows the median and 95th percentile of the distance measurements compared to the actual distance, while Figure 9(b) shows the error in distance calculation using the median of the distance measurements. The motes were within line of sight, as is the case in all experiments presented.

We see that the error in distance estimation is overall at most 26cm and on average about 3cm. It is worth noticing that in most cases error falls within the margin of measurement error, given that mote size is 2.5cm and motes are manually placed. Also, error is smaller for distances between 0.5m and 9m and increases for smaller or larger distances. This is because of the calculation of THRESH-A. In small distances threshold THRESH-A cannot be properly calculated as sound measurements tend to reach the maximum and



(a) Estimated distance vs real distance



(b) Error in estimated distance

Figure 9. Localization in one dimension.

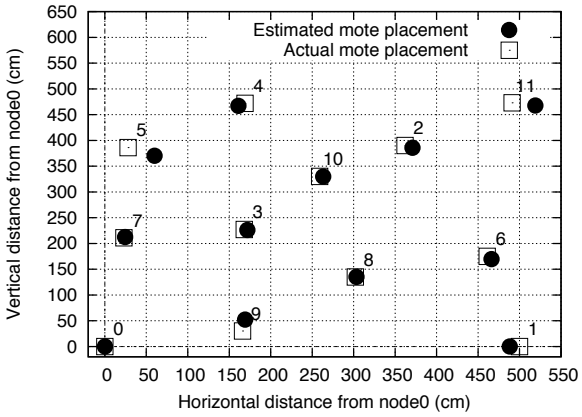


Figure 10. Localization in two dimensions.

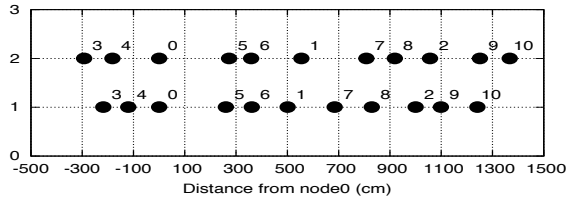


Figure 11. Moving reference point.

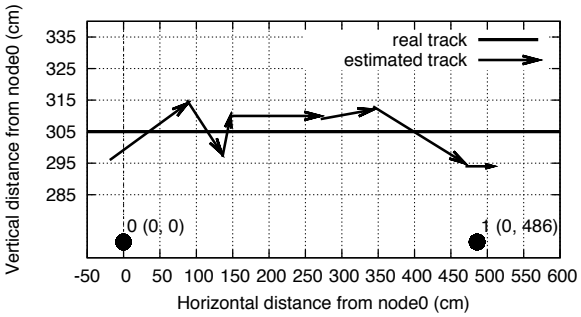


Figure 12. Tracking motion.

minimum values the ADC can detect, thus INC_{MAX} gets a maximum value which is not representative of the sound amplitude. For this reason, for small distances, mapping sound ToF to distance should not be done with a linear mapping function. In our experiments a binomial function for small values of sound ToF was more effective for small distances. However, we do not include these results here.

3.2.2. 2-D Localization. We use 2-D setup to easily examine the accuracy of the localization scheme in a multi-sensor setup. We place twelve motes in arbitrary positions on the floor of a $5 \times 5m^2$ square room. Nodes 0 and 1 are denoted as reference sensors and are equipped with a buzzer. Figure 10 shows the actual distances between sensors in this setup, as well as estimated distances. Nodes 0 and 1 produce a sound pulse in turn once every 20s for a total period of 10min. Every time an emitter produces a sound all other motes estimate their distance from it and forward their timestamps to a gateway mote (not shown). Our results show that the median localization error is 8cm whilst the maximum localization error is 36cm and corresponds to the location of node 5. This is mostly due to error in distance estimation from node 1.

3.2.3. Multiple Reference Points. Next, we examine how localization can occur over longer distances by using multiple reference points. We use a 1-D setup, with eleven sensors. Three sensors, 0, 1, and 2, are denoted as reference points and are placed 5m apart. In between, before, and after, we place eight sensors, covering in total 14.6m. Localization runs every 5min. Figure 11 shows our results. Horizontal line 1 shows the actual position of motes, while line 2 shows the estimated locations, using the closest reference point for each sensor. We see that error in localization reaches up to 1.5m. This is a direct consequence from the fact that in this experiment all motes (emitters and listeners) were powered by batteries, which entails that soon enough emitters could not draw enough current for sound pulse emission, thus producing sound pulses with distorted features.

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 to 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 12 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.

4. Related work

4.1. Clock Synchronization

Previous work has proposed various protocols for clock synchronization in wireless sensor networks.

Our work is similar to the flooding time synchronization protocol (FTSP) [14]. FTSP achieves high accuracy by operating at MAC-Layer and eliminating non-deterministic delays by means of calibration through multiple timestamps. In contrast to our protocol, which requires a single master broadcasting synchronized timestamps, in FTSP all synchronized nodes broadcast synchronized timestamps for multi-hop synchronization. FTSP achieves average synchronization error on Mica2dot platform 4μs and maximum synchronization error 12μs. Similar to FLASH, it uses linear regression to compensate for the drift in between resynchronization periods. The main shortcoming of FTSP is that it is hardware dependent, since the transmission delay time is predefined, thus requiring calibration for each type of antenna chipset used. Moreover, they measure error in clock synchronization using RF messages that may introduce inaccuracy in error estimation, whereas we use an external mechanism that relies on a common (global) clock.

Delay measurement time synchronization (DMTS) [16] is similar to RBS on the sender side and FTSP on the receiver side, with the difference that the 32kHz crystal is used as the timing source, providing a lower precision yet a simpler synchronization scheme that can be used in applications with lower synchronization accuracy. DMTS achieves synchronization accuracy of 32μs.

In reference broadcasting synchronization (RBS) [17] a node transmits a signal and the rest of the nodes use that signal as a time reference to record their local time at the moment of reception, estimating their time offset with each other. The main advantage of this approach is that it eliminates all non-deterministic procedures at the transmitter, even though it does not manage to do the same with receivers. Its main disadvantage 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 reference mote, increasing network load. RBS achieves average synchronization accuracy of 30μs.

In time synchronization protocol for sensor networks (TPSN) [13] a sensor node acting as root synchronizes the rest of the network nodes that are organized into a spanning tree. In TPSN each synchronization act between nodes comprises of a round-trip message exchange during which both the propagation delay and the relative offset of the two clocks are estimated. The main advantage of TPSN, similar to our approach, is that timestamping takes place at the MAC-Layer, thus eliminating some of the larger non-deterministic delays in message transmission. Its main disadvantage is the $2 \cdot N$ messages required for synchronizing N sensor nodes. Moreover, it does not provide a mechanism to compensate for the clock drift of nodes. TPSN achieves average synchronization accuracy of less than 18μs.

4.2. Sound-based Localization

Previous work on localization focuses on GPS, TDoA measurement between RF and ultrasound signals [5], [6] or received RF signal properties to infer distance [9], [10]. None of these however, is appropriate for sensor networks.

The authors in [1] use acoustic sounds and ToA measurements for localization. However, they assume that Mica2dot motes are not adequate for efficient sound detection in software and use PDAs attached to the sensors to achieve localization. PDAs and motes are all synchronized in a global time-scale using RBS with each node broadcasting a synchronization packet every 10s. The motes act as sound emitters and the PDAs that detect the sound send the corresponding time series over 802.11 to the PDA that is connected to the emitter. Sound time series are correlated using a sliding correlator in search of the point of maximum correlation. Experimental results in a lab environment of $8m \times 10m$ show that average error in localization is 11.5cm. We achieve a similar error, however, without the use of PDAs.

Azimi-Sadjadi et al. [2] connect wireless sensor nodes to an FPGA and form 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 over FTSP. Whenever a sound is detected in the environment, the hardware requires a synchronized timestamp from the connected mote and then sends this timestamp along with the acoustic time series concerning the sound to a base station, where time series are cross-correlated and distance from the sound source is calculated using TDoA. Experimental results show that the error in localization is up to 50cm for distances up to 20m.

Lopes et al. in [4] develop a system using PDAs and acoustic sensors. PDAs produce sound pulses that are sensed by sensors connected to desktops that maintain synchronized clocks. Synchronization with the PDAs is temporarily achieved by sending an RF signal that notifies each PDA to produce a sound and each sensor to listen for the sound. Then TDoA is measured and translated into distance. In their setup they use six sensors in a $23m \times 9m$ space and examine

measurements of the three sensors that are closest to the sound source. They show a median error of localization from 5cm to 74cm, depending on the position in room.

Finally, in Calamari [3] TDoA between RF and audible sound are used to infer distance in range up to 130cm. However, in that case sound is sensed with the use of special hardware tone detectors and calibration of every sensor against all other sensors is required before deployment, otherwise localization error can be up to 30cm.

Overall, our method is the first to achieve high accuracy, fine-grained localization without the use of specialized support, excessive resources, or frequent calibration and demonstrate results in a real setup.

5. Conclusions and Future Work

In this work, we present a scheme for sensor localization that achieves fine-grained localization, at the level of tens of centimeters, primarily for indoor applications, and does not require any support from the environment, additional infrastructure, sensor hardware support, or excessive sensor resources. Our approach relies on consistent acoustic sound pulse detection over high-precision clock synchronization via RF communication. We discuss our method in detail and present an implementation on a real sensor network that consists of Mica2dot motes. We present measurements on various setups and show that our approach manages to synchronize clocks with an accuracy of 5μs and estimate relative sensor positions within 11 cm for distances up to 7m. Moreover, range estimation is consistent for distances more than 10m and it allows for multi-hop reference points for longer ranges. Future work should explore how our localization approach can be adapted to more demanding environments, such as in the presence of obstacles between nodes, high noise-levels, temperature and humidity variations, and outdoor environments. Overall, our work shows that accurate, fine-grained localization can be achieved with low-cost, off-the-shelf sensor nodes. We believe that this is an important step towards using sensor networks in real-life applications.

6. Acknowledgments

This work was made possible with support from the Institute of Computer Science, Foundation for Research and Technology - Hellas (FORTH-ICS). We are thankful to Michalis Ligerakis for his valuable help with power and sound peripheral issues and the members of the CARV laboratory at FORTH-ICS for the discussions during the course of this work.

References

[1] 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*, pp. 214–219, Sep. 2002.

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

[4] 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.*, vol. 10, no. 2, pp. 38–50, 2006.

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

[6] 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)*, New York, NY, USA, 2001, pp. 166–179.

[7] A. Ward, A. Jones, and A. Hopper, "A new location technique for the active office," *IEEE Personnel Communications*, vol. 4, no. 5, pp. 42–47, October 1997. [Online]. Available: http://dx.doi.org/10.1109/98.626982

[8] R. Want, A. Hopper, a. Veronica Falc and J. Gibbons, "The active badge location system," *ACM Trans. Inf. Syst.*, vol. 10, no. 1, pp. 91–102, 1992.

[9] 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)*, New York, NY, USA, 2005, pp. 1–12.

[10] 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)*, vol. 2, pp. 775–784 vol.2, 2000.

[11] 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)*, New York, NY, USA, 2004, pp. 174–181.

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

[13] 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)*, New York, NY, USA, 2003, pp. 138–149.

[14] 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)*, New York, NY, USA, 2004, pp. 39–49.

[15] J. Elson and K. Römer, "Wireless sensor networks: a new regime for time synchronization," *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 1, pp. 149–154, 2003.

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

[17] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," *SIGOPS Oper. Syst. Rev.*, vol. 36, no. SI, pp. 147–163, 2002.