Sonoloc: Scalable positioning of commodity mobile devices

Viktor Erdélyi
MPI-SWS
Saarland Inf. Campus, Germany

Trung-Kien Le
National Institute of Informatics
Tokyo, Japan

Bobby Bhattacharjee
University of Maryland
College Park, MD, USA

Peter Druschel
MPI-SWS
Saarland Inf. Campus, Germany

Nobutaka Ono
National Institute of Informatics
Tokyo, Japan

ABSTRACT

We present Sonoloc, a mobile app and system that allows a set of co-located commodity smart devices to determine their relative positions without local infrastructure. Sonoloc enables users to address each other based on their relative positions at events like meetings, talks, or conferences. This capability can, for instance, aid spontaneous communication among users based on their relative position (e.g., in a given section of a room, at the same table, or in a given seat), facilitate interaction between speaker and audience in a lecture hall, and enable the distribution of materials, crowdsensing, and feedback collection based on users' location. Sonoloc can position any number of devices within acoustic range with a constant number of chirps emitted by a self-organized subset of devices. Our experimental evaluation shows that the system can locate up to hundreds of devices with an accuracy of tens of centimeters using up to 15 audio chirps emitted by dynamically selected devices, in actual rooms and despite substantial background noise.

CCS CONCEPTS

• **Information systems → Location based services**; *Sensor networks*; • **Human-centered computing → Mobile computing**;

ACM Reference Format:
Viktor Erdélyi, Trung-Kien Le, Bobby Bhattacharjee, Peter Druschel, and Nobutaka Ono. 2018. Sonoloc: Scalable positioning of commodity mobile devices. In *MobiSys '18: The 16th Annual International Conference on Mobile Systems, Applications, and Services, June 10–15, 2018, Munich, Germany*. ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3210240.3210324

1 INTRODUCTION

Mobile apps and services enable organizers of events such as conferences, meetings, concerts, or other events to disseminate information and collect feedback in real time. Likewise, these apps allow event attendees to use such apps to network, exchange commentary, and share information during or after the event.

A key challenge for such apps is how to enable parties to address each other in a convenient and usable manner, particularly when among new acquaintances or strangers. Most current apps and services require users to identify communication partners by their (nick-)names, e-mail addresses, phone numbers, or OSN IDs. However, requiring users to manually exchange contact details is awkward and impractical at large events, and many individuals find it difficult to remember the names of everyone they met at an event.

A naïve approach would be to automatically exchange names, contact details, and portraits among participants, either by having users' devices broadcast this information via short-range radio, or, in case of an organized event, by publishing the information on the event web site. However, this approach is questionable from a privacy perspective.

A better approach would be to name communication partners by their relative spatial position at a given time. For instance, it is easier to refer to a group of people as "those sitting at the same table" than by enumerating their names, and many individuals tend to remember the relative seating positions of people they met at a meeting or dinner table more easily than their names.

In addition, spatial naming naturally enables communication scenarios where the position of a user is all that matters, and not their identity. For instance, one might wish to disseminate different versions of an exam to test-takers in adjacent seats or disseminate notes to everyone present at a given table. Likewise, one might wish to collect feedback about the audiovisual experience in a concert hall based on the seat a user occupies, tag crowdsourced pictures or videos with the position from where they were taken, or dispatch responders based on the position of an individual who has a request or reports an emergency.

In this paper, we present Sonoloc, a system that allows a set of commodity smart devices within acoustic range to determine their *relative position map* without local infrastructure. The map can be displayed on the device, allowing the user to identify communication senders or receivers by their relative position at the time the map was created. By providing a (possibly ephemeral) network address and an optional nickname along with their relative position, participating devices can communicate conveniently and without exchanging any long-term identifiers or personally identifying information at all.

Sonoloc is able to determine a relative position map among hundreds of commodity smart devices in a large room with a constant number of audio chirps emitted by a dynamically selected subset of devices. The protocol proceeds in several rounds. Initially, a set of randomly selected devices emit a sequence of audio chirps. Meanwhile, *all* devices listen to these transmissions. The devices then compute their positions using differences in chirp arrival times. First, the transmitters' positions are computed, and then the remaining devices are positioned relative to the set of transmitters.

MobiSys '18, June 10–15, 2018, Munich, Germany

ACM ISBN 978-1-4503-5720-3/18/06.
https://doi.org/10.1145/3210240.3210324

By sharing their measurements, the devices can tentatively place themselves on a map. Next, additional transmitters are selected strategically based on their positions on the tentative map, which emit more chirps to refine the map in the next round.

The paper makes the following contributions. (1) We present Sonoloc, a system that can, upon the request of one user, provide a relative position map of up to hundreds of commodity smart devices within audio range, without requiring local infrastructure.[1] Unlike naïve audio ranging approaches, which would require $\Theta(n)$ chirps to position n devices, Sonoloc requires an audio chirp from only a small subset of the participating devices. The key is a novel, multi-stage iterative transmitter selection and device localization protocol that can position any number of devices using chirps from a small number of strategically selected devices. (2) We present results of extensive simulations with many device layouts, and we experimentally evaluate an Android Sonoloc prototype with up to 15 Motorola Nexus 6 phones and 100 Raspberry Pi-based passive devices in different rooms and with different background noise levels. Our results show that Sonoloc can position hundreds of devices with an accuracy of tens of centimeters with no more than 15 audio chirps, and with signal-to-noise levels as low as −20 dB.

In the rest of this paper, we discuss related work in Section 2 and present the overall Sonoloc protocol in Section 3. The audio signal processing pipeline is described in Section 4 and we present the localization algorithms in Section 5. Results of an experimental evaluation based on simulations and real-world experiments are presented in Section 6. We conclude in Section 7.

2 RELATED WORK

Sonoloc produces a relative map of devices within acoustic range. Such a map can be derived if pairwise distances between each device are available, but in Sonoloc, some of these distances must be inferred since not all devices transmit. Hence, Sonoloc builds on two broad areas of work: positioning using inter-device distance measurements (Section 2.1) and localization based on TOA/TDOA measurements (Section 2.3). Alternative approaches based on local infrastructure are discussed in Section 2.2.

Distances can be measured using radio and audio signals. Table 1 presents a comparison of representative protocols that use RF, acoustic ranging (AR), or both. Measurements based on radio frequency (RF) signals are unobtrusive but, as the table shows, tend to be relatively inaccurate [8, 16, 17] or require specialized hardware [39, 36, 34, 35] and/or changes to infrastructure [36, 34, 35]. In contrast, measurements based on audio signals can be accurate on the order of centimeters, do not require deployment of special hardware or infrastructure, but emit audible sounds.

Compared to prior AR-based systems that do not require specialized infrastructure, Sonoloc achieves similar median accuracy. However, in contrast to prior work, which either did not evaluate maximum error or have several meters of maximum error, Sonoloc has maximum errors of around 0.5 m. This is required to position users relative to each other correctly in practice. Moreover, Sonoloc can position hundreds of devices with a constant number of chirps.

[1] Our prototype relies on cellular or Wi-Fi access, but this is not fundamentally required.

2.1 Relative device positioning

The BeepBeep protocol [28] measures the pairwise distance between two devices by serially emitting a chirp on each device's speaker, listening on the microphones of each device, and computing the distance based on the observed signal arrival times. For pairwise measurements, BeepBeep has an average accuracy of around 2 cm, and it is the basis of pairwise transmitter distance measurements in Sonoloc.

Determining the relative positions of devices using pairwise BeepBeep to support file sharing was proposed in [30]: this method requires $\Theta(n)$ measurements to map n devices, which is impractical in a large venue with hundreds of devices for the following reasons. First, each transmission comes with audible sound, and hundreds of transmissions would become very disruptive to human users. Second, each transmission takes on the order of seconds, and, since transmissions need to be sufficiently separated in time for good signal-to-noise ratio, hundreds of transmissions would lead to very large delays. Sonoloc instead uses a constant number of transmitters *regardless of the number of devices in the room.*

Ping-Pong [13] determines the relative positions of multiple devices based on pairwise distances measured via acoustic ranging. Like the original BeepBeep, Ping-Pong requires $\Theta(n)$ measurements to position n devices, but each measurement emits a musical note. Ayllón et al. [3] combine AR ranging and inertial sensor readings to estimate the relative position of nearby devices. All devices need to emit chirps, limiting the method's scalability. The FAR system [40], designed for mobile gaming, uses AR to continuously track the distance between two phones with a median error of 2 cm. Measurements are quick (100ms) but are pairwise only. A protocol to maintain the relative 3D position of two phones using AR cues and inertial sensors is described in Qiu et al. [29]. In contrast, Sonoloc performs a one-time positioning of many devices with a small, constant number of chirps.

2.2 Infrastructure-based localization

Device localization protocols determine absolute geographic coordinates, and are important indoors where GPS is often unavailable. Sufficiently accurate indoor localization could be used to create a relative position map as well. However, existing techniques are either not accurate enough [8], require changes to deployed infrastructure [16, 34, 36], specialized local infrastructure like audio beacons with known locations [18, 19, 25], access to pre-defined radiomaps [38], or specialized hardware [34, 36, 1, 2].

Radiomaps measure signal strength from different APs at many points within indoor locations, and existing techniques [38] can use such maps to interpolate device locations. Other techniques can use the angle of arrival information from multiple antennas at APs to achieve sub-meter resolution [34, 36]. The accuracy of these methods varies based on density and construction of base stations (number of stations and antennas per station), changes to APs (requires re-computation of radiomap), scattering environment, and may vary with indoor occupancy.

Centaur [26] combines RF and AR based techniques for indoor localization, using the distances measured using AR as geometric constraints for Bayesian inference on the RF profiles of the participating devices. A similar approach is taken in Liu et al. [24],

Protocol	Medium	Accuracy	Max. error	Tested Range	Custom HW	# Meas. for map	Meas. speed	Notes
PinPoint [39]	RF	>1m avg.	4.3 m	44.5 m	All	$\Theta(n)$	fast	non-Wi-Fi HW for fine grained timestamps
ArrayTrack [34]	RF	0.26m med.	–	floor	All	N/A	fast	6+ antennas per AP, custom client HW
Synchronicity [35]	RF	1.6m med.	–	floor	APs	N/A	fast	Requires SW/HW change at APs
ToneTrack [36]	RF	0.9m med.	5 m	floor	All	N/A	fast	Custom devices and APs
Horus [38]	RF	0.5m med.	2.9 m	floor	-	N/A	fast	Requires radiomaps
EZ Localization [8]	RF	2-6m med.	20 m	floor	-	N/A	fast	Integrates GPS, server support
SpotFi [16]	RF	0.4m med.	12 m	floor	-	N/A	fast	AP SW change
Ubicarse [17]	RF	0.4m med.	–	floor	-	N/A	fast	localizes non-RF objects via vision
BeepBeep [28]	AR	0.02m avg.	–	12 m	-	$\Theta(n)$	slow	pairwise distance only, line-of-sight results
Ping-Pong [13]	AR	0.03-0.2m	–	2.8 m	-	$\Theta(n)$	slow	Error device-size dependent
FAR [40]	AR	0.02m med.	–	<2 m	-	$\Theta(n)$	fast	pairwise distance only, 12Hz updates
Liu et al. [25]	AR	0.1-0.2m med.	0.35 m	7.3 m	Beacons	$\Theta(n)$	slow	Requires deploying anchor nodes. Limited eval. with 1 phone in 2 locations.
Centaur [26]	RF+AR	0.6m med.	3.5 m	floor	-	$\Theta(n)$	slow	Combines Wi-Fi and acoustic
ALPS [19]	RF+AR	0.1-0.2m med.	3.2 m	72 m^2	Beacons	$\Theta(n)$	fast	Requires ultrasound HW beacons
Akiyama et al. [2]	AR+Light	0.01m	–	<5 m	Beacons	$\Theta(n)$	fast	Requires sync. LED beacons
SmartSLAM [12]	RF+Inert.	1.6m (66%), 2.7m (95%)	–	floor	-	N/A	medium	2Hz updates, sensor fusion incl. radiomaps
Sonoloc	AR	0.06m med.	0.5 m	17.8 m	-	O(1)	slow	Position map with up to 15 signals

Table 1: Survey of representative positioning and location protocols using radio frequency (RF) or acoustic ranging (AR). "Fast" measurements require <100ms, whereas "slow" ones require >1s. For solutions that do not transmit device-to-device signals (such as infrastructure-based solutions), the number of measurements is not applicable.

where BeepBeep-based AR ranging informs Wi-Fi profile based localization. Centaur also introduces two new AR measurement techniques: EchoBeep, which performs well even without a line-of-sight between two devices, and DeafBeep, which works with speaker-only devices. Both these techniques can be directly integrated with Sonoloc.

Finally, SmartSLAM [12] performs simultaneous localization and mapping (SLAM) using Wi-Fi fingerprinting, inertial sensors and compass on smartphones. The system moves between different sensor fusion algorithms depending on available prior knowledge, to minimize CPU load and energy consumption; its accuracy is on the order of several meters.

Sonoloc instead positions devices relatively and accurately, without requiring infrastructure support.

2.3 TOA/TDOA localization algorithms

Devices can be positioned with respect to each other without infrastructure support, instead based solely on peer measurements. Joint speaker and microphone localization based on the measured distances between speakers and microphones has been studied in previous work [27, 22, 20, 21]. This problem is known as *time-of-arrival (TOA)-based joint source and sensor localization*. The method from [21] is used in Sonoloc to localize transmitters. A second type of localization, microphone localization based on distance differences and known positions of speakers, is referred to as *time-difference-of-arrival (TDOA)-based source localization* [33]. Closed form solutions exist [31, 32, 7, 37, 5]. Sonoloc uses a variant of the spherical interpolation method [33] to localize passive (non-transmitting) devices.

3 SONOLOC DESIGN

In this section, we describe Sonoloc's overall design and its iterative protocol for computing a relative position map among the set of participating devices. Since Sonoloc relies on acoustic ranging to measure distances and distance differences among devices, it needs to be able to detect its audio signals accurately in noisy environments. The signal design and processing used by Sonoloc for this purpose are described in Section 4.

The key challenges in designing a practical system are robustness to background noise, independence from specialized hardware and local infrastructure, and scalability to many devices with reasonable delay and minimal inconvenience to users, e.g., via a small number of audible signals.

Given a set of transmitters and the measured signal arrival times at all devices, Sonoloc computes a relative position map in three steps. First, from the signal arrival times, Sonoloc computes the pairwise distances among the transmitters; for each passive device (i.e., a device not chosen as a transmitter), it computes the distance difference between that device and each pair of transmitters. Second, Sonoloc computes a map of transmitters consistent with the measured pairwise transmitter distances. Third, it places the passive devices into the same map consistent with the measured distance differences. Section 5 describes the techniques we use to compute these maps from the signal arrival times, in the presence of inaccurate measurements.

3.1 Assumptions

Sonoloc makes the following assumptions:

- Participating users carry a smart device that has support for Bluetooth, a Wi-Fi or cellular network connection, a

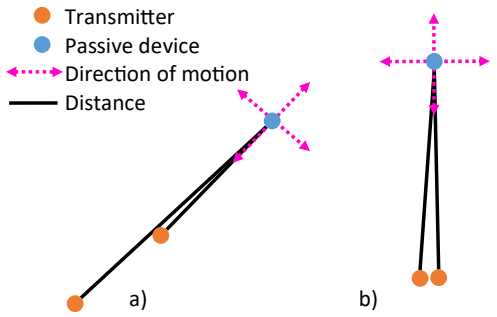

Figure 1: Poor transmitter choice examples

microphone and a speaker. All modern smart phones have these capabilities.

- The distance between the speaker(s) and microphone(s) for each phone is known. For this purpose, the app provider can run a simple online database with the measured distances for all common device models.
- All participating devices are within audible distance from each other and users place their phones into a relatively unobstructed position (e.g., by holding them up in the air or by placing them on the table).
- Phones do not move while Sonoloc performs measurements. This assumption is consistent with Sonoloc's intended usage, where a user initiates the system when participants are seated in an auditorium or at a table.
- Because Sonoloc computes a two-dimensional position map, we assume that the vertical (z) offsets among participating devices are small. We evaluate the impact of larger z-offsets on Sonoloc's accuracy in Section 6.
- All participating devices run the Sonoloc app. Any device running the Sonoloc app can initiate the protocol upon user request, by contacting nearby devices that also run the app via Bluetooth Low Energy.

3.2 Strawman Sonoloc protocol

Consider the following strawman protocol:

1. Select t transmitters randomly from the participating devices.
2. Emit a chirp from each transmitter in sequence while all devices listen.
3. Determine the chirp arrival times at each device (Section 4).
4. Compute the position map from the chirp arrival times measured at all devices (Section 5).

This protocol produces position maps with good median accuracy under reasonable audio conditions. However, occasionally, the protocol produces high positioning errors due to a poor choice of transmitters. What is a poor choice of transmitters? Consider the simple examples depicted in Figure 1. Recall that a passive device determines its relative position based on the measured *difference* in distance from each pair of transmitters.

Example a): Two transmitters and a passive device lie roughly along a straight line. Although the difference in the distance to the two transmitters is large, if the passive device moved in any direction, it would initially observe very small changes to this difference. Therefore, accurately positioning p based on this distance difference requires very high precision measurements.

Example b): Two transmitters are very close, relative to the distance from a passive device. If the passive device moved in any direction, it would initially observe very little change in the distance to the transmitters. Again, accurately positioning the passive device requires very high precision measurements.

The accuracy of distance measurements, however, is limited by the audio sampling rate (at 48 kHz, sound travels 7.13 mm during one sample, assuming ambient temperature of 20 °C), line-of-sight obstructions that may lead to the detection of a reflected signal, and z-offsets. Given these limitations, the transmitters must be well distributed in both dimensions of the space covered by all devices, so that each passive device finds enough pairs of transmitters with large angles between them. The challenge is how to choose a good distribution of transmitters in the absence of a map, which we are trying to produce as the output of the protocol.

3.3 Iterative Sonoloc protocol

We meet this challenge with an iterative transmitter selection protocol. We start with a small set of randomly chosen transmitters and compute a map. This map will not be accurate but provides an approximation sufficient to choose an additional transmitter strategically. In subsequent iterations, a newly chosen transmitter emits an audio signal, and we re-run the localization to refine the map. (We *do not* re-select the full set of transmitters in each iteration.) The iteration continues until the desired accuracy is reached.

Choosing the size of the initial transmitter set T_{init} involves a tradeoff between overall protocol delay and the quality of additional transmitter choices. A smaller T_{init} improves our ability to select more transmitters strategically. A larger T_{init} improves the quality of the initial map (which is the basis for the choice of the next transmitter), and reduces overall protocol delay, because T_{init} can be selected in one shot. Empirically, we found $|T_{init}| = 6$ to be a good choice. See [11] for additional details.

The goal of choosing additional transmitters in each iteration of the protocol is to try and have transmitters along the full range of x and y coordinates occupied by passive devices. Towards this end, we tried a number of different heuristics to choose a set of additional transmitters. The following works well empirically.

In each iteration (other than the first, when transmitters are chosen randomly), we compute the convex hull of the transmitters in the current map. Then, we select a passive device outside the hull that will, if chosen as a transmitter, increase the area covered by the convex hull the most. This step is illustrated in Figure 2. Intuitively, by maximizing the area covered by the convex hull of transmitters, we seek to include most passive devices in the hull. A passive device inside the hull is "surrounded" by transmitters, thus avoiding bad transmitter choices like those shown in Figure 1. If all passive devices lie inside the convex hull already, we choose additional transmitters from the set of passive devices randomly.

The full Sonoloc protocol is shown in Algorithm 1. We evaluate the number of transmitters (I) required for good accuracy empirically in Section 6.

Input: D: set of participating devices
Input: I: desired number of transmitters to use
Output: position map
Select 6 initial transmitters randomly from D;
All devices start recording;
Initial transmitters emit chirps sequentially;
while *#transmitters* < *I* **do**
  Collect measurements, compute map;
  Compute convex hull of transmitters;
  if *all devices in D inside hull* **then**
    Select new transmitter randomly from D;
  else
    Select new transmitter that maximizes area of convex hull;
  end
  New transmitter emits chirp;
end
All devices stop recording;

Algorithm 1: Sonoloc protocol

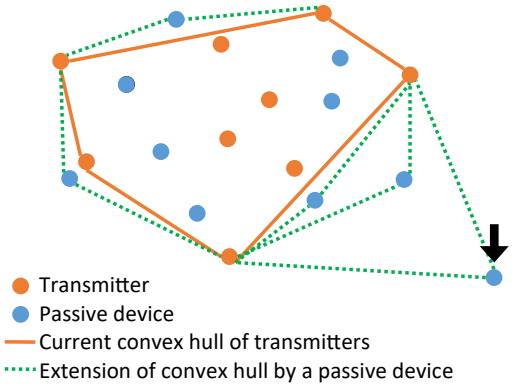

Figure 2: Illustration of transmitter selection based on the convex hull. In the example, the bottom right passive device (marked with an arrow) is selected as an additional transmitter.

3.4 Map ambiguity and orientation

Sonoloc computes a 2D relative position map consistent with observed distances and distance differences. Note that a map corresponding to a view from above the 2D plane and one from below the 2D plane are both consistent with the observed measurements. Sonoloc relies on the initiating user to resolve this ambiguity and "flip" the map if needed.[2] Each user can also rotate the 2D map around its center to match their current orientation in the plane. See [11] for a discussion of (semi-)automated ways of resolving the map-flipping and rotation problems.

[2]If the initiator's location happens to be on a line of symmetry within the map, it may not be obvious to her if the map is "upside down". In this case, the initiator can ask another user to flip the map as needed.

4 SIGNAL DESIGN AND DETECTION

This section describes the audio signal design and detection mechanisms used by Sonoloc.

4.1 Signal design

A suitable audio signal must have three properties: 1) The signal should be distinguishable from background noise. 2) It should be feasible to transmit and sense the signal using commodity smartphone hardware. 3) The signal should not be similar to a time-shifted version of itself. Otherwise, the correlation function used to detect a received signal would generate multiple peaks, some of which could obfuscate the true start of the signal.

Following standard practice in radar and sonar applications, we use a linear chirp [9, 10, 23]. Because of its flat and wide spectrum, it produces a sharp peak in autocorrelation. We use a linear upward sweep from 1 kHz to 17 kHz with a length of 500 ms.

4.2 Signal detection

Signal detection consists of two stages: (1) computing the correlation of the recorded waveform and the reference signal, and (2) selecting the "correct" peak in the correlation output to detect the start of transmission. Unfortunately, simply choosing the highest peak is not sufficient because peaks can occur due to reflections and interference. We describe the two stages of signal detection in turn.

4.2.1 Computing correlation. For computing correlation, we apply the Generalized Cross-Correlation with Phase Transform (GCC-PHAT) algorithm [15, 6, 4]. GCC-PHAT is a filtered version of the usual cross-correlation that includes a frequency-shaping component to improve performance by making the correlation peak "sharper". GCC-PHAT consists of a cross-correlation phase and a whitening filter that flattens the spectrum.

It is possible to compute the correlation by applying FFT over the entire received signal. However, applying the whitening filter over the entire received signal can lead to spurious peaks in the cross-correlation because the whitening filter has a very long impulse response. Instead, we apply the whitening filter by using Short-Time Fourier Transform (STFT) with a bounded frame length of 128 samples. Figure 3 shows an example comparing GCC-PHAT+STFT to standard correlation and standard GCC-PHAT. The audio waveform is a phone recording of a signal emitted by another phone in the presence of loud background music. In this example, regular cross-correlation (or GCC-PHAT with full FFT) does not produce a clear spike in the time domain, but GCC-PHAT with STFT does.

4.2.2 Signal detector. In real world scenarios, the correlation signal can have many peaks due to reflections, and we must choose one as the start of reference. We need to detect the direct (shortest) path signal, because the time-of-flight of reflected signals does not match the physical device distance. We expect the corresponding correlation peak to be the earliest one. Reflected signals appear after the direct path signal, and, due to direct-path obstructions, they may correlate with the reference signal more strongly than the direct path signal. Thus, we cannot simply choose the highest correlation peak.

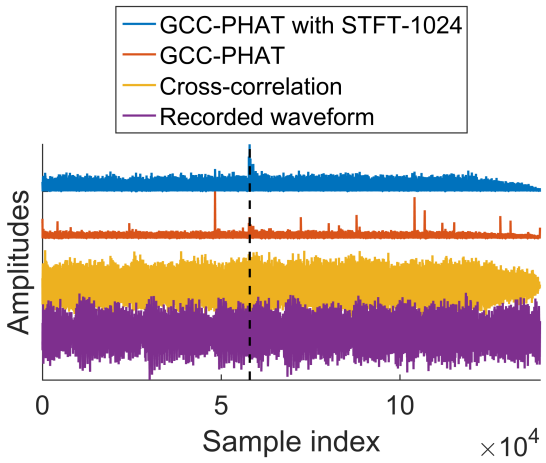

Figure 3: Audio recording of signal and background music, and different correlation signals. The signal starts at the dashed vertical line

Symbol	Meaning
d_{xy}	Distance from x's speaker to y's microphone
t_x	Global time at which x transmitted a signal
r_{xy}	Global time at which x's signal was received at y
f	Sampling rate (Hz)
c_s	Speed of sound (343.2 m/s @ 20 °C)

Table 2: Notation; times are expressed in terms of a (hypothetical) global clock.

Sonoloc's detector is based on CFAR [14], with parameters determined empirically. We estimate the noise levels, during the entire recording and during the past 1000 samples, and set two thresholds, global and local, each 15 dB above the corresponding estimated noise level. Additionally, since correlation peaks come sometimes in small bursts with a ramp-up and ramp-down, the algorithm tries to get to the top of the ramp by requiring that the peak is a local maximum among all samples within +/-60 samples of the peak. We choose the first peak in the correlation signal that satisfies all 3 conditions (global threshold, local threshold, and local maximum) as the start of the received signal. It is possible that the correlation signal does not contain such a prominent peak; in this case, the detector signals a failure.

5 LOCALIZATION

This section describes how Sonoloc computes a position map from the measured times of arrival determined by the signal detector from Section 4. We begin with a description of how Sonoloc determines distances and distance differences from the signal arrival times. Then, we describe the algorithm for determining transmitter locations from their pairwise differences. We close with the algorithm for determining the positions of passive devices from their measured distance differences to the transmitters. We use the notation in Table 2. The relevant distances are illustrated in Figure 4.

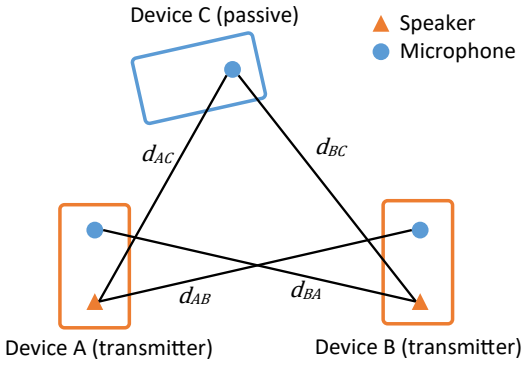

Figure 4: Distance notation

5.1 Estimating distances and distance differences

Using the acoustic signal processing techniques described in Section 4, each device determines the time at which an audio signal (including its own transmitted signal) captured by its microphone starts. At first, the arrival time of a received signal is expressed in terms of the receiver's local audio sampling clock, i.e., it is an index in a list of received audio samples. To express it in *seconds*, as required by the equations below, the sample index must be divided by the audio sampling rate in Hz.

Sonoloc uses these independent arrival timestamps to derive two types of possible observations: (i) distances between the speaker in one transmitter and the microphone in another transmitter, and (ii) distance differences given by a triple of two speakers in two transmitters and a microphone in a passive device. The protocol only uses arrival time differences of signals captured by the same device, therefore the audio sample clocks need not be synchronized.

5.1.1 Estimating distances. For notational convenience, we assume a hypothetical global clock for the time equations below. However, none of the computations at any device require a global clock. Instead, each device only computes *intervals* based on their local recorded timestamps. Assume that a set of transmitters have emitted sounds. We generalize the algorithm described in Peng et al. [28] to estimate, for all pairs of transmitters, the distance between the speaker in one transmitter and the microphone in another transmitter.

Given a signal emitted by a speaker in a device A at time t_A according to a global clock, the following two equations hold for its reception at devices A and B, respectively:

$$r_{AA} = t_A + \frac{d_{AA}}{c_s} \quad \text{and} \quad r_{AB} = t_A + \frac{d_{AB}}{c_s}. \tag{1}$$

Note that, for convenience, we are defining the time at which A's signal is received at B in terms of when A sends its message. Devices can be localized without knowing the value of t_A. ($t_A - t_B$, used later, can be determined even if t_A and t_B are unknown.)

Analogous equations hold for a signal emitted by a speaker in a device B: r_{BB} and r_{BA}. We eliminate t_A and t_B so that we obtain:

$$d_{AB} + d_{BA} = d_{AA} + d_{BB} + c_s(r_{BA} - r_{AA} + r_{AB} - r_{BB}) \tag{2}$$

If d_{AA} and d_{BB} are small compared to d_{AB} and d_{BA}, we can use $(d_{AB} + d_{BA})/2$ to estimate both d_{AB} and d_{BA}, the distance between A's speaker and B's microphone, and the distance between B's speaker and A's microphone, respectively.

If a signal fails to be detected, the associated pairwise distance among transmitters will be missing. In this case, we can estimate the missing value as follows: for all triples of transmitters involving the pair with the missing distance, compute bounds based on the triangle inequality. However, signal detection failures did not occur in our experiments.

5.1.2 Estimating distance differences. Suppose A and B are transmitters (as above) and C is a passive device that locally timestamps the audio signals from A and B. Our goal is to derive the distance difference $\tau_{AB|C} = d_{AC} - d_{BC}$. Similarly to the equations in (1), it is clear that

$$r_{AC} = t_A + \frac{d_{AC}}{c_s} \quad \text{and} \quad r_{BC} = t_B + \frac{d_{BC}}{c_s}. \tag{3}$$

Thus,

$$\tau_{AB|C} = d_{AC} - d_{BC} = c_s(r_{AC} - r_{BC} - (t_A - t_B)) \tag{4}$$

Device C can directly measure the time difference $r_{AC} - r_{BC}$ locally. The quantity $t_A - t_B$ can be derived at device B as follows:

$$t_A - t_B = \frac{d_{BB} - d_{AB} + c_s(r_{AB} - r_{BB})}{c_s} \tag{5}$$

Note that the derivation requires d_{AB}, which we can estimate using $\frac{(d_{AB}+d_{BA})}{2}$ as long as the distance between devices is much larger than the distance between speaker and microphone on the same device.

Finally, in order to reduce the impact of noisy distance differences (due to inaccurate signal detection) on the localization accuracy, Sonoloc filters the estimated distance differences based on the triangle inequality. According to the triangle inequality, $\forall A, B \in Transmitters : \forall C \in Passive : |\tau_{AB|C}| < d_{AB}$. We compute the set of device triples that violate the triangle inequality by a given tolerance factor F:

$$\begin{aligned}\{(A, B, C) \mid A, B \in Transmitters \wedge C \in Passive \\ \wedge |\tau_{AB|C}| \geq F \cdot d_{AB}\}\end{aligned} \tag{6}$$

For each such triple, we ignore the associated distance difference constraint in the subsequent localization.

In sensitivity experiments, we found that this filtering is required for good localization accuracy. The value of F needs to be low enough to filter out high-error distance difference measurements that negatively affect the results. Empirically, we determined that $F = 1.2$ is a good choice. See [11] for details.

5.2 Computing coordinates

Based on the measurements described above, the localization phase produces coordinates for all participating devices in Sonoloc. We use two types of localization: (1) transmitter localization via joint speaker-microphone localization based on the distances estimated in Section 5.1.1, and (2) passive device localization via microphone localization based on the distance differences estimated in Section 5.1.2 and the coordinates of the transmitters computed in (1).

More precisely, for a device A, let $\mathbf{s}_A$ be the position of A's speaker, and let $\mathbf{m}_A$ be the position of A's microphone. Since our goal is to localize *devices* (and not speakers and microphones), we define the position of a device A as follows. If A is a transmitter, we define its position as the average of the coordinates of its speaker and microphone, i.e.,

$$pos_A = \frac{1}{2}(\mathbf{s}_A + \mathbf{m}_A). \tag{7}$$

If A is a passive device, we define its position as the coordinates of its microphone, $\mathbf{m}_C$. The first localization estimates $\mathbf{s}_A, \mathbf{m}_A$ for all transmitters A based on *estimated* distances given for all pairs of transmitters A, B based on Section 5.1.1:

$$\hat{d}_{AB} \simeq d_{AB} = \|\mathbf{s}_A - \mathbf{m}_B\| \tag{8}$$

where $\|\cdot\|$ is the Euclidean distance. The second localization estimates $\mathbf{m}_C$ for all passive devices C based on *estimated* distance differences given for all pairs of transmitters A, B with respect to C based on Section 5.1.2:

$$\hat{\tau}_{AB|C} \simeq \tau_{AB|C} = d_{AC} - d_{BC} = \|\mathbf{s}_A - \mathbf{m}_C\| - \|\mathbf{s}_B - \mathbf{m}_C\| \tag{9}$$

(note that $\tau_{AB|C} = -\tau_{BA|C}$), and the known (estimated) positions

$$\{\hat{\mathbf{s}}_A \mid A \in Transmitters\} \tag{10}$$

from the first localization.

5.2.1 Localizing transmitters. Given estimated pairwise distances $\{\hat{d}_{AB} \mid A, B \in Transmitters\}$, we use a least-squares criterion to choose the *estimated* speaker and microphone coordinates of transmitters $\{\hat{\mathbf{s}}_A, \hat{\mathbf{m}}_A \mid A \in Transmitters\}$ so that it minimizes the following objective function:

$$\sum_{A,B \in \text{Transmitters}} \left(\hat{d}_{AB} - \|\hat{\mathbf{s}}_A - \hat{\mathbf{m}}_B\|\right)^2. \tag{11}$$

In general, the objective function of (11) is non-convex over the coordinates of speakers and microphones; finding the optimal solution is prohibitively expensive. We use an iterative method based on existing work; this method is sensitive to the "starting point" for the optimization — an arbitrary initialization of coordinates can get "trapped" into a local minimum far from the global one. Using existing work described in [21], we compute a closed-form solution for the constraint (8) and use this solution to initialize the optimization of (11).

Iterative refinement. The closed-form solution for transmitter positioning assumes that the measured distances are precise, i.e.,

$$\forall A, B \in Transmitters : \hat{d}_{AB} = \|\mathbf{s}_A - \mathbf{m}_B\|. \tag{12}$$

In practice, the measured distances are noisy, leading to inaccuracies in the computed transmitter coordinates. Fortunately, as long as the noise is not "too" large and the closed-form solution is near the global minimum, they can be improved further using existing work [27].

Given an initial solution, the method iterates using an auxiliary function for nonlinear least-squares optimization to improve the solution. The quality of the final solution depends on the initial solution; fortunately, in our case, the closed-form method gets us "close". Our Sonoloc implementation stops the refinement after 10 iterations. Sensitivity experiments suggest that additional iterations have negligible impact. See [11] for more details.

Sonoloc transmitter localization algorithm. A summary of the transmitter localization in Sonoloc is shown in Algorithm 2.

Input: Estimated pairwise distances:
$\hat{D} = \{(A, B, \hat{d}_{AB}) \mid A, B \in Transmitters\}$.
$\forall A \in Transmitters : \hat{d}_{AA} = 0$.
Output: Estimated positions: TrPos
$= \{(A, \hat{\mathbf{s}}_A, \hat{\mathbf{m}}_A) \mid A \in Transmitters\}$.
TrPos = Apply closed-form solution [21] for $\hat{D}$;
TrPos = Iteratively apply update rule [27] for TrPos and $\hat{D}$;

Algorithm 2: Sonoloc transmitter localization.

5.2.2 Localizing passive devices. In Sonoloc, N speakers in N transmitters emit audio chirps, and these sounds are recorded by the N microphones in N transmitters and the M microphones in M passive devices. The speaker and microphone coordinates of transmitters are estimated using the closed-form (+iterative) algorithm described above. Next, we describe how Sonoloc estimates the coordinates of the M remaining microphones.

Based on the distance difference measurements described in Section 5.1.2 and the known coordinates of the transmitter speakers, the estimation of the position of a passive device's microphone maps precisely to the *source localization from range-difference measurements* problem studied in [32, 33, 37]. Crucially, the constraints corresponding to the different microphones are entirely *independent*, in that there is no dependence between microphones, their locations, recording, or computation. This property allows Sonoloc to scale. As long as microphones can record the sound samples with sufficient fidelity and undertake the procedure listed below, the system scales to an arbitrary number of passive devices, regardless of location. This property is useful because, e.g., in big lecture halls, hundreds of devices can be within audio range. In addition, the computation for each passive device is independent and can be executed in parallel.

For a passive device C, $\mathbf{m}_C$ can be estimated as follows. Given the estimated speakers positions of all transmitters $\{\hat{\mathbf{s}}_A \mid A \in Transmitters\}$ and the observed distance differences $\{\hat{\tau}_{AB|C} \mid A, B \in Transmitters\}$, we choose the *estimated* position $\hat{\mathbf{m}}_C$ that minimizes

$$\sum_{A,B \in \text{Transmitters}} \left[\hat{\tau}_{AB|C} - \left(\|\hat{\mathbf{s}}_A - \hat{\mathbf{m}}_C\| - \|\hat{\mathbf{s}}_B - \hat{\mathbf{m}}_C\|\right)\right]^2. \quad (13)$$

The optimization (13) has a non-convex cost function of only two coordinates of $\hat{\mathbf{m}}_C$. Thus, (13) is simpler than (11), and the existing closed-form solutions work well for the optimization (13). Sonoloc uses the closed-form solutions described in [33] to solve the optimization (13). Note that [33] describes localizing *speakers* based on known microphone locations, but in Sonoloc, we localize microphones based on known speaker positions. Our problem is symmetric in the sense that distance differences can be defined between pairs of speakers and one microphone as we do, or, between pairs of microphones and one speaker as described in [33].

Iterative refinement. As before, the closed-form solution assumes that the measured distance differences are accurate. In practice, the differences are noisy, leading to inaccurate output. In order to improve the quality of the solution given by the closed-form method, we use an iterative algorithm based on prior work [27]. Since the work in [27] shows that this solution is never worse than the closed-form solution, we always use the coordinates provided by the iteratively refined solution.

Our implementation stops the refinement after a maximum of 2000 iterations or when the improvement in the value of the objective function falls below a threshold where Sonoloc's overall accuracy is not affected anymore. Our sensitivity experiments suggest that the refinement is required for good accuracy; no more than 2000 iterations were needed in any of our experiments. See [11] for more details.

Noise type	Lowest robust SNR
Synthetic Gaussian	1.53 dB
Real noise (Cafeteria)	−22.6 dB
Real noise (Talk)	−24.5 dB
Real noise (Street)	−34.1 dB
Real noise (Poster session)	−24.2 dB

Table 3: SNR levels for reliable signal detection

6 EVALUATION

In this section, we present results of our experimental evaluation of Sonoloc. The evaluation has three parts. First, we evaluate the performance of Sonoloc's signal detection under different types of noise and signal-to-noise ratios. Second, we present results of extensive simulations that allow us to explore many more room and device layouts than we could reasonably cover in real experiments. It also enables us to explore the impact of factors like the selection and number of transmitters, inaccuracies in detected signal arrival timestamps, and z-offsets. Finally, we present real-world experiments in different rooms with a prototype implementation running on up to 15 Motorola Nexus 6 smartphones and up to 100 Raspberry Pi 3 Model B units as passive devices.

6.1 Signal detection

Detecting the audio chirp signal in a recorded waveform is essential for accurate localization. We evaluate the Sonoloc signal detector under controlled conditions by mixing the clean reference signal with various types of noise at different signal-to-noise ratios (SNR). We are using synthetic and recorded noise for our evaluation.

For synthetic noise, we use Gaussian random noise with a zero mean and unit standard deviation. For real noise evaluations, we recorded background noise in various environments, including street noise from an open window, a talk (presentation), a poster session, and a cafeteria during lunch hour. The recorded noise recordings are several minutes long; in each experiment, we mix in the signal at a different random offset within the noise recording. During mixing, the amplitude of the noise is adjusted to achieve a desired SNR.

We run the signal detector on the noisy recordings and compare the detected signal position with the known ground-truth position of the signal in the recording. Table 3 shows the minimal SNR required for reliable detection with different types of noise. Detection

is considered reliable if, during at least 100,000 trials, there were no detection failures and the signal was detected within three audio samples of the true position in each trial.

As we can see, Sonoloc's detector can reliably detect signals in real noise down to SNRs below −20 dB. This robustness is possible because the acoustic spectrum of real noise is typically limited. Such a spectrum contains strong frequencies affected by loud noise, and some weak frequencies largely unaffected by the noise. Given such a situation, the whitening filter selectively enhances the weaker frequency bands that are unaffected by the noise. In contrast to realistic noise, synthetic Gaussian noise is full-band, so the technique we just described does not help.

6.2 Positioning simulations

Next, we show results of our extensive simulations to explore Sonoloc's accuracy in positioning devices systematically under many different device layouts, transmitter selections, various levels of signal detection inaccuracy, z-offsets, etc. We begin with a description of the assumptions underlying our simulations.

Device layouts. We simulated a large number of room shapes and device layouts likely to arise in practice. In all cases, devices are placed in a plane and minimally 50 cm apart from each other, reflecting a typical private space people tend to keep.[3]

Random: Pick random coordinates for the devices within either a 20 m × 20 m quadratic space or a circular space with a 10 m radius.

Clustered: Pick a given number (e.g. 5) of cluster centroids at a random location within a 20 m × 20 m space at least 1 m apart. Then, populate each cluster with 4-30 additional devices at random locations with a distance from the centroid exponentially distributed with a mean of 1 m.

In the following regular device layouts, we perturb each device's nominal location by choosing a uniformly random 2D position up to 10cm away from the nominal position to account for typical variations in device positions relative the users' nominal position (e.g. in a given seat).

Concentric circles: 100 devices are laid out in two concentric circles with radii 9 and 18 m, respectively.

Cafeteria: Devices are arranged around 36 rectangular tables with 8 devices each; table centers are arranged in a grid 3 m apart.

Banquet: Devices are arranged around 10 round tables with 12 devices each. Tables are placed at random locations within a 25 m × 25 m room.

Small restaurant: Devices are arranged around 6 rectangular tables 2.5 meters from each other with 6 devices each in a small 5 m × 10 m rectangular room.

Lecture hall: Devices are placed randomly into 20 rows of 20 seats each with a separation of 0.5 m between rows and seats.

Conference room: Devices are placed on a long, rectangular table, where seats are separated by 0.5 m and the two long edges are 2 m apart.

The location of a device is considered identical with the position of its speaker. To simulate the effect of different device orientations, we obtain a device's microphone location by adding to the speaker's location a 3D vector with a random orientation and a magnitude equivalent to the known speaker-microphone distance of the device. To simulate the effect of signal detection errors, we perturb the actual arrival time by a uniform random number of audio samples in the range [−1, 15].

We evaluate the positioning accuracy in terms of the *interpoint distance error (IDE)*: for each pair of devices, the absolute difference between the true distance and the distance based on the estimated positions between the devices' centers, in meters. We consider a device's center the halfway point between its speaker and microphone.

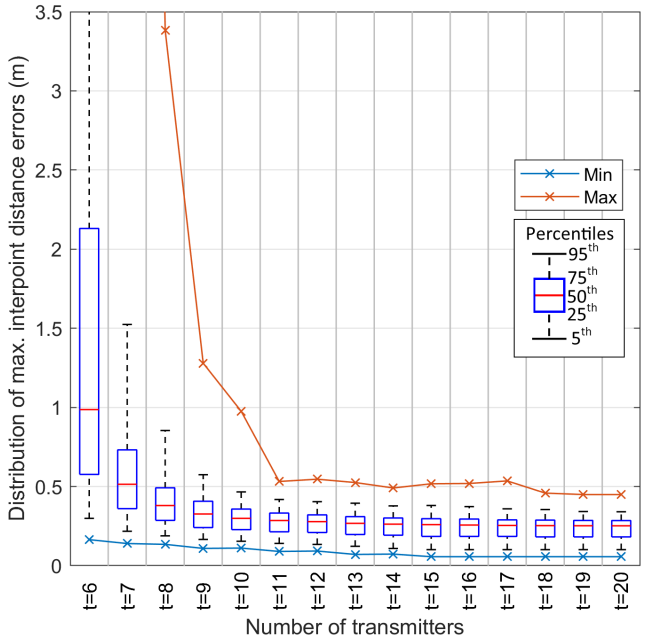

Figure 5: Max IDE vs. #transmitters; 500 scenarios. Percentiles 5, 25, 50, 75, and 95 shown.

Overall positioning accuracy. Figure 5 shows the statistic of the maximal IDE in each individual experiment as a function of the number of transmitters used, over a total of 500 simulated sample runs of the Sonoloc protocol. Each run used a different device layout chosen in equal proportions from the types described above. The results show that with 11 or more transmitters, the maximal IDE is around 0.5 m, and, in half of the experiments (see median), the max. IDE is below 0.3 m. This is a very strong result, considering the conservative assumptions in our simulations (challenging device layouts, inaccurate signal detection, etc.).

Sensitivity. We have performed extensive simulations to explore the sensitivity of Sonoloc's positioning accuracy to various factors. Due to space limitations, we summarize the results here. More details are available in [11].

Signal detection inaccuracy: As discussed above, we assume uniform random simulated signal detection errors in the range [−1, 15] samples, as this choice yields the best agreement with our empirical results. To understand the sensitivity, we also explored other error

[3]People can stand closer in crowded circumstances like a subway train, but it is not clear if Sonoloc would normally be used in this situation.

distributions. Assuming perfectly accurate signal detection, the median and maximal IDE is below 0.06 m and 0.33 m, respectively. Assuming large detection errors in the range $[-25, 25]$, on the other hand, the median and maximal IDE increases to 0.11 m and 1.55 m, respectively. We conclude that Sonoloc can tolerate frequent but modest signal detection errors well.

Z offsets: Sonoloc does 2D localization, but in practice, devices may have vertical offsets. To explore the impact of such offsets on Sonoloc's positioning accuracy, we assigned each device a uniform random z-offset. For a z-offset in the range $[-0.5, 0.5]m$, the median IDE is unaffected and the maximal IDE increases to 0.87 m. However, for z-offsets in the range $[-1, 1]m$, the median and maximal IDEs increase to 0.1 m and 1.95 m. We conclude that Sonoloc can tolerate z-offsets of up to 0.5 m with acceptable accuracy.

Ambient temperature: The speed of sound varies with air temperature. Sonoloc's calculations assume a temperature of 20 °C and our simulations show that Sonoloc can tolerate temperature deviations of up to 5 °C with modest impact on IDEs. However, larger temperature deviations require compensation. Future smart devices will likely have thermometers, making it possible to perform this compensation automatically.

Device layout: Figure 5 includes results with all different device layout types. We also ran separate simulations with the individual device layout types described above. In general, device layouts can be more or less challenging, depending on the extent to which they constrain transmitter selection. The easiest layouts are the small random layouts (15 devices), followed by a small clustering layout (3 clusters, 15 devices), then the conference room (30 devices) and small restaurant layout (30 devices). The most challenging are the large random layouts (100 devices), the lecture hall (100 devices), and the cafeteria layout (288 devices). The latter are responsible for the largest IDEs shown in Figure 5.

Transmitter selection algorithm: As discussed in Section 3, a good choice of transmitters is important to achieve high accuracy. We compared different transmitter selection algorithms on many device layouts of the types described above. To get a sense of the best and worst possible algorithm, we also tried a large number of random transmitter selections on a given device layout and recorded the best and worst of N random selections, for $N = 100$. In terms of maximal IDE, Sonoloc's transmitter selection algorithm achieved 0.57 m, while the best of 100 achieved 0.35 m. The worst of N selection has a maximal IDE of almost 11 m, underscoring the importance of transmitter selection. Also, a single random transmitter selection resulted in a maximal IDE of 8.4 m, which shows that intelligent transmitter selection is important.

6.3 Prototype implementation

The Sonoloc prototype app was implemented on Android 7.1.1 and relies on a JDK 8-based backend localization server; apps installed on participating devices send their recorded sound samples to the server, which analyses the signals, computes and disseminates the devices' position map. The app communicates with the server over Wi-Fi, and with other nearby devices running the app over Bluetooth. The server performs the signal detection and localization using MATLAB and Java code. We use MATLAB Compiler SDK to interface with MATLAB implementations of the localization algorithms.

We chose a server-based implementation because it allowed us to conveniently conduct many experiments under controlled conditions, and to experiment with different algorithms on the same recorded data. An alternate implementation could process the audio signals on the mobile devices, sending only the time differences to the location server.

We use a 48 kHz audio sampling rate for both transmission and recording. We transmit the chirp on one of the speakers (the top speaker on the Motorola Nexus 6), and we mute the other speaker. Sonoloc records signals on all available microphones. However, combining the signal from multiple microphones is difficult, because the microphones are in different positions and their relative distance to a given transmitter depends on the phone's orientation, which is unknown. Instead, Sonoloc dynamically chooses to use the microphone that recorded the "best" signal.

A transmitter A needs to determine the difference in the arrival times of its own signal and the signal from a transmitter B. It chooses the microphone that reports the earliest arrival of B's signal and then uses the same microphone to detect its own signal. Thereby, Sonoloc can take advantage of the fact that one microphone may have an unobstructed path to B while other microphones don't.

Using different microphones on the same device for estimating distances to different devices adds noise to distance estimates. We accept this extra error since using the *best* microphone for a given measurement proves to be necessary in practice. Consider an alternative where a single microphone, however chosen, is used on a device. Unfortunately, if this microphone cannot be used for distance measurements from even a single transmitter (which can happen due to physical barriers or other environmental factors), then we receive no distance estimate for this pair. One option is to ignore this measurement. However, since transmitter localization requires *all* pairwise distances, doing so would cause a single poor measurement to result in the transmitter being discarded by all devices!

A passive device C instead needs to determine the TDOA of the signal from two transmitters A and B. Moreover, since it determines its location based on the TDOA of all transmitter pairs, it needs to use the same microphone for all transmitter pairs. Therefore, on a given device, Sonoloc uses the microphone that detects a signal from the largest number of the transmitters.

In contrast to transmitter localization, passive device localization can work with any (sufficiently large) number of distance differences. Thus, it is less risky to choose a single microphone (e.g., the one that provides the best signal quality for the majority of measurements) and ignore an individual distance difference whenever this "majority" microphone cannot be used. When a distance difference is ignored, we still lose some accuracy, but, unlike losing a transmitter that affects *all* passive devices, only a single passive device is affected. Since the associated risk is smaller, Sonoloc uses the majority microphone approach for passive device localization.

6.4 Live experiments

We conducted live experiments with our Sonoloc prototype running on Motorola Nexus 6 phones. To enable experiments with

large numbers of devices at reasonable cost, we also ran Sonoloc on Raspberry Pi (RPi) single-board computers with Plantronics Audio 300 microphones and Sabrent AU-MMSA USB audio adapters. We experimented in different rooms and several buildings on our campus, and under different noise conditions.

The noise conditions were *quiet*: only background noise from HVAC and other equipment (38 dB–40 dB); *speech*: playback of recorded speech (49 dB–55 dB); and *music*: playback of recorded music (50 dB–62 dB). (Measured with a PeakTech 5055 sound level meter with A-weighting.) The phones emitted chirps at the highest possible volume setting using only one channel. During the experiments, the phones and RPis were not moved.

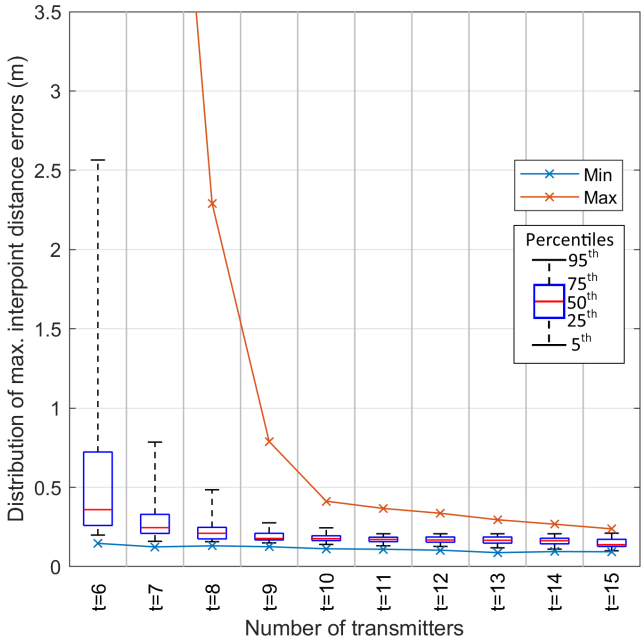

Figure 6: Max IDE vs. #transmitters; various rooms; up to 15 phones. Percentiles 5, 25, 50, 75, and 95 shown.

Our first set of experiments were conducted with up to 15 phones in a small 10-person conference room, a medium-sized 60-seat classroom, a 100-seat classroom, and a 180-seat classroom. In each case, we distributed phones throughout the rooms. The phones were held above the seats and tables using smartphone holders. There was an unobstructed line-of-sight between most but not all pairs of phones. Pairwise distances among phones ranged from 0.7 to 2.7 m in the small room, from 0.6 to 4.3 m in the medium-sized room, from 0.94 to 9.6 m in the 100-seat classrooms, and 0.6 to 13.1 m in the 180-seat classroom.

In each case, we conducted experiments with silence, speech, or music replayed. Figure 6 shows the statistic of the maximal IDE in each individual experiment as a function of the number of transmitters used across 36 protocol runs in the different rooms, and different background noise. The results show that Sonoloc achieves maximal IDEs of less than 0.5 m for 10 or more transmitters. The presence of background noise did not significantly affect the IDEs (not shown).

We also conducted experiments in a large, 180-seat lecture hall with 15 phones and 100 RPis. In the lecture hall, the pairwise distances between devices range from 0.5 to 17.8 m. Since the RPis have no speakers, they cannot be chosen as transmitters. Therefore, we conducted the experiment as follows. We randomly chose 115 out of the 180 seats in the large lecture hall to be "occupied". We ran a simulation with the exact room layout and seat occupancy to let Sonoloc choose a set of transmitters from the 115 occupied seats. Then we placed the phones into the seats that were chosen as transmitters, we filled the remaining seats with RPis, and ran the protocol as normal, except that we constrained the transmitter selection to within the set chosen in the simulation.

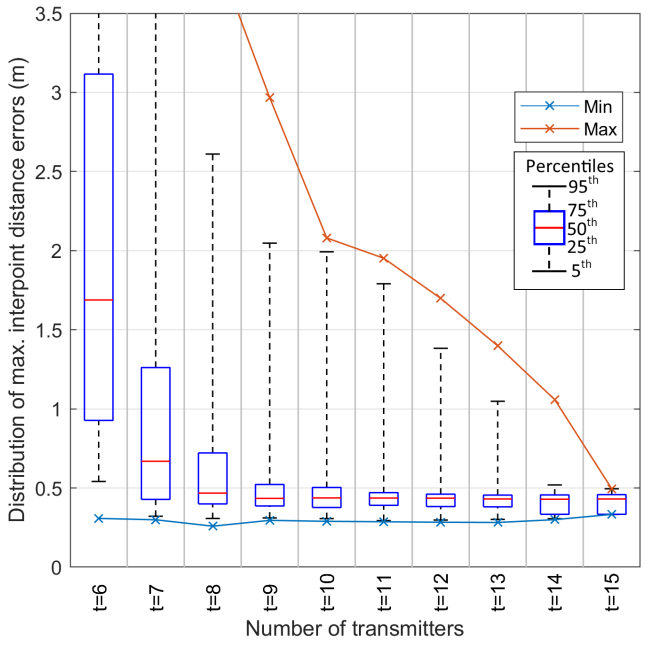

Figure 7: Max IDE vs. #transmitters; 180-seat lecture hall; 115 devices. Percentiles 5, 25, 50, 75, and 95 shown.

Figure 7 shows the statistic of maximal IDEs with 115 devices in the lecture hall, as a function of the number of transmitters used. The results include seven protocol runs with different types of background noise. The results show that Sonoloc can position 115 devices to a maximal IDE of 0.5 m with only 15 audio chirps under real-world conditions. Again, the presence of background noise did not significantly affect the IDEs (not shown).

Compared to the results shown in Figure 6, where only phones were used, Sonoloc requires 15 transmitters to achieve a maximal IDE under 0.5 m in this experiment, and the median and average IDE are higher. Further investigation showed that the higher IDEs are associated with the position of the RPis and not the phones. These devices exhibit higher signal detection errors and larger sample clock drift, most likely due to the lower quality of the microphones and sound cards we used with these devices. These additional errors would not occur in practice, where all devices are assumed to be smartphones.

# Devices	Average	Std. Dev.	Max.
20	3.2	0.5	3.8
100	10.4	5.7	22.9
500	48.5	13.4	68.6

Table 4: Localization runtimes in seconds.

Operational range: Sonoloc's range is limited by background noise, audio signal attenuation, and the properties of speakers and microphones. In our experiments, Sonoloc worked reliably at distances of up to 18 m.

6.5 Sonoloc runtime overhead

In this section, we evaluate Sonoloc's runtime overhead. First, we consider the processing overhead of the signal detector, which includes the execution times of the correlation and the peak detection stages. We measured the runtime of the signal detector on 242 recorded audio waveforms during one of the live experiments (lecture hall, background music). Since the detection overhead depends largely on the length of the recorded audio waveform, we normalized the runtime to the length of the recordings.

On a Dell Precision T1700 computer with 16 GB of RAM and an Intel(R) Xeon(R) CPU E3-1246 v3 @ 3.5 GHz, the signal detector took a minimum, median, maximum of 0.16, 0.19, and 0.3 seconds, respectively, per second of recorded audio waveform.

The runtime of the localization algorithm depends on the number of transmitters and the number of passive devices. We generated 10 random device layouts with 20, 100, and 500 devices in a 20 m × 20 m square space, and ran the algorithm on each layout with 15 transmitters. In Table 4, we report average and maximal runtime, as well as the standard deviation across 10 sequential runs of the algorithm.

The focus in our prototype implementation so far has been on accuracy and robustness. There are numerous, unexplored opportunities for optimizing the localization runtime. In addition, both the transmitter localization and the passive device localization can be parallelized. For instance, the localization of different passive devices is independent and can be trivially executed in parallel.

The end-to-end delay for a complete run of the protocol is ultimately limited by the following requirements: (1) For good SNR, the chirps emitted by the initial transmitter set must be separated in time sufficiently so that strong echoes of a transmission do not overlap with subsequent transmissions; (2) the time between subsequent transmissions must allow for signal analysis, localization, and selection of an additional transmitter based on the latest map.

6.6 Discussion: Alternative implementations

As mentioned above, the Sonoloc prototype performs signal processing and localization on a server for maximal control, reproducibility, and the ability to experiment with different algorithms and parameters on recorded audio waveforms. Here, we discuss implementation options for an actual deployment.

A natural choice is to perform the audio signal processing on the phones and upload only the inter-arrival times between the recorded signals to a server, which orchestrates the protocol, selects transmitters and performs localization. This architecture reduces privacy concerns, because no recorded audio leaves a phone. It also reduces network traffic at the expense of some additional computation on the phones.

It is also possible to perform the protocol entirely on the phones. Here, the initiator's device orchestrates the protocol and performs the initial transmitter selection. It collects the signal inter-arrival times from the participating phones via Bluetooth, performs the transmitter localization, and broadcasts the results. Each passive device then localizes itself and send its coordinates to the initiator devices, which selects an additional transmitter and iterates. Since, in this case, the compute-intensive localization is performed on the phones, there will be somewhat longer delays and higher energy consumption. However, this mode could be used as a fallback in the case where an Internet connection is unavailable.

7 CONCLUSION

We presented and evaluated Sonoloc, a mobile app and system that can determine the relative 2D positions of hundreds of mobile devices within audio range with a small, constant number of audio chirps. The system assumes that devices have a speaker, microphone, and Bluetooth or Wi-Fi capabilities, but does not require any other local infrastructure. Our experimental evaluation, which included detailed simulations and live experiments with over one hundred devices in a large space, shows that, under a wide range of conditions, Sonoloc can position devices with median/maximal position errors of around 0.06 m and 0.5 m, respectively.

We designed and parameterized Sonoloc to have worst-case accuracy of no more than 0.5 m, even under challenging conditions. In this respect, Sonoloc represents the state-of-the-art in positioning performance without any sort of additional infrastructure, such as radiomaps or fixed beacons, or special hardware, such as lasers or fine-grained timestamping. We believe Sonoloc is accurate enough to enable new classes of data dissemination, local sharing, and social applications based on spontaneous position maps obtained using regular devices in everyday situations.

ACKNOWLEDGMENTS

We would like to thank our shepherd, Heather Zheng, and the anonymous reviewers for their insightful and helpful comments and suggestions. Additionally, we thank Rose Hoberman and Matthew Lentz for their feedback on earlier drafts of the paper.

The work was supported in part by the European Research Council (ERC Synergy imPACT 610150), the German Science Foundation (DFG CRC 1223), a Grant-in-Aid for Scientific Research (A) (KAKENHI Grant Number 16H01735) from Japan Society for the Promotion of Science (JSPS), and NSF Awards CNS 1526635 and CNS 1314857.

REFERENCES

[1] T. Akiyama, M. Sugimoto, and H. Hashizume. "Light-synchronized acoustic TOA measurement system for mobile smart nodes". In: *Proceedings of IPIN*. 2014. DOI: 10.1109/IPIN.2014.7275557.

[2] T. Akiyama, M. Sugimoto, and H. Hashizume. "SyncSync: Time-of-arrival based localization method using light-synchronized acoustic waves for smartphones". In: *Proceedings of IPIN*. 2015. DOI: 10.1109/IPIN.2015.7346958.

[3] D. Ayllón et al. "Indoor Blind Localization of Smartphones by Means of Sensor Data Fusion". In: *IEEE Transactions on Instrumentation and Measurement* (2016). DOI: 10.1109/TIM.2015.2494629.

[4] M. Azaria and D. Hertz. "Time delay estimation by generalized cross correlation methods". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* (1984). DOI: 10.1109/TASSP.1984.1164314.

[5] M.S. Brandstein, J.E. Adcock, and H.F. Silverman. "A closed-form location estimator for use with room environment microphone arrays". In: *IEEE Transactions on Speech Audio Processing* (1997). DOI: 10.1109/89.554268.

[6] G. C. Carter. "Coherence and time delay estimation". In: *Proceedings of the IEEE* (1987). DOI: 10.1109/PROC.1987.13723.

[7] Y.T. Chan and K.C. Ho. "A simple and efficient estimator for hyperbolic location". In: *IEEE Transactions on Signal Processing* (1994). DOI: 10.1109/78.301830.

[8] Krishna Chintalapudi, Anand Padmanabha Iyer, and Venkata N. Padmanabhan. "Indoor Localization Without the Pain". In: *Proceedings of MobiCom*. 2010. DOI: 10.1145/1859995.1860016.

[9] C. E. Cook. "Pulse Compression-Key to More Efficient Radar Transmission". In: *Proceedings of the IRE* (1960). DOI: 10.1109/JRPROC.1960.287599.

[10] Charles Cook. *Radar Signals: An Introduction to Theory and Application*. Electrical science series. Elsevier Science, 2012. ISBN: 9780323146302.

[11] Viktor Erdélyi. *Scalable positioning of commodity mobile devices using audio signals*. Tech. rep. MPI-SWS-2018-002. MPI-SWS, 2018.

[12] Ramsey M. Faragher and Robert K. Harle. "Towards an Efficient, Intelligent, Opportunistic Smartphone Indoor Positioning System". In: *Journal of the Institute of Navigation* (2015). DOI: 10.1002/navi.76.

[13] Jorge Herrera and Hyung-Suk Kim. "Ping-pong: Using smartphones to measure distances and relative positions". In: *The Journal of the Acoustical Society of America* (2013). DOI: 10.1121/1.4831185.

[14] The MathWorks Inc. *Constant False Alarm Rate (CFAR) Detection*. URL: https://www.mathworks.com/help/phased/examples/constant-false-alarm-rate-cfar-detection.html (visited on 09/28/2017).

[15] C. Knapp and G. Carter. "The generalized correlation method for estimation of time delay". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* (1976). DOI: 10.1109/TASSP.1976.1162830.

[16] Manikanta Kotaru et al. "SpotFi: Decimeter Level Localization Using WiFi". In: *Proceedings of SIGCOMM*. 2015. DOI: 10.1145/2785956.2787487.

[17] Swarun Kumar et al. "Accurate Indoor Localization With Zero Start-up Cost". In: *Proceedings of Mobicom*. 2014. DOI: 10.1145/2639108.2639142.

[18] Patrick Lazik and Anthony Rowe. "Indoor Pseudo-ranging of Mobile Devices Using Ultrasonic Chirps". In: *Proceedings of SenSys*. 2012. DOI: 10.1145/2426656.2426667.

[19] Patrick Lazik et al. "ALPS: A Bluetooth and Ultrasound Platform for Mapping and Localization". In: *Proceedings of SenSys*. 2015. DOI: 10.1145/2809695.2809727.

[20] T.-K. Le and N. Ono. "Closed-form and Near closed-form solutions for TOA-based joint source and sensor localization". In: *IEEE Transactions on Signal Processing* (2016). DOI: 10.1109/TSP.2016.2569465.

[21] T.-K. Le and N. Ono. "Closed-form solution for TDOA-based joint source and sensor localization in two-dimensional space". In: *Proceedings of EUSIPCO*. 2016. DOI: 10.1109/EUSIPCO.2016.7760473.

[22] T.-K. Le and N. Ono. "Numerical Formulae for TOA-based Microphone and Source Localization". In: *Proceedings of IWAENC*. 2014. DOI: 10.1109/IWAENC.2014.6954002.

[23] Nadav Levanon and Eli Mozeson. *Radar signals*. John Wiley & Sons, 2004. ISBN: 978-0-471-47378-7.

[24] Hongbo Liu et al. "Push the Limit of WiFi Based Localization for Smartphones". In: *Proceedings of MobiCom*. 2012. DOI: 10.1145/2348543.2348581.

[25] Kaikai Liu et al. "Towards accurate acoustic localization on a smartphone". In: *Proceedings of INFOCOM*. 2013. DOI: 10.1109/INFCOM.2013.6566822.

[26] Rajalakshmi Nandakumar, Krishna Kant Chintalapudi, and Venkata N. Padmanabhan. "Centaur: Locating Devices in an Office Environment". In: *Proceedings of MobiCom*. 2012. DOI: 10.1145/2348543.2348579.

[27] N. Ono et al. "Blind alignment of asynchronously recorded signals for distributed microphone array". In: *Proceedings of WASPAA*. 2009. DOI: 10.1109/ASPAA.2009.5346505.

[28] Chunyi Peng et al. "BeepBeep: A High Accuracy Acoustic Ranging System Using COTS Mobile Devices". In: *Proceedings of SenSys*. 2007. DOI: 10.1145/1322263.1322265.

[29] Jian Qiu et al. "On the Feasibility of Real-time Phone-to-phone 3D Localization". In: *Proceedings of SenSys*. 2011. DOI: 10.1145/2070942.2070962.

[30] Jun-Wei Qiu et al. "A D2D relative positioning system on smart devices". In: *WCNC*. 2014. DOI: 10.1109/WCNC.2014.6952645.

[31] H.C. Schau and A.Z. Robinson. "Passive source localization employing intersecting spherical surfaces from time-of-arrival differences". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* (1987). DOI: 10.1109/TASSP.1987.1165266.

[32] J. Smith and J. Abel. "Closed-form least squares source location estimation from range-difference measurements". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* (1987). DOI: 10.1109/TASSP.1987.1165089.

[33] P. Stoica and J. Li. "Lecture Notes - Source Localization from Range-Difference Measurements". In: *IEEE Signal Processing Magazine* (2006). DOI: 10.1109/SP-M.2006.248717.

[34] Jie Xiong and Kyle Jamieson. "ArrayTrack: A Fine-Grained Indoor Location System". In: *Proceedings of NSDI*. 2013. ISBN: 978-1-931971-00-3. URL: https://www.usenix.org/conference/nsdi13/technical-sessions/presentation/xiong.

[35] Jie Xiong, Kyle Jamieson, and Karthikeyan Sundaresan. "Synchronicity: Pushing the Envelope of Fine-Grained Localization with Distributed MIMO". In: *Proceedings of HotWireless*. 2014. DOI: 10.1145/2643614.2643619.

[36] Jie Xiong, Karthikeyan Sundaresan, and Kyle Jamieson. "ToneTrack: Leveraging Frequency-Agile Radios for Time-Based

Indoor Wireless Localization". In: *Proceedings of Mobicom*. 2015. DOI: 10.1145/2789168.2790125.

[37] L. Yang and K.C. Ho. "An approximately efficient TDOA localization algorithm in closed-form for locating multiple disjoint sources with erroneous sensor positions". In: *IEEE Transactions on Signal Processing* (2009). DOI: 10.1109/TSP.2009.2027765.

[38] Moustafa Youssef and Ashok Agrawala. "The Horus WLAN Location Determination System". In: *Proceedings of MobiSys*. 2005. DOI: 10.1145/1067170.1067193.

[39] Moustafa Youssef et al. "PinPoint: An Asynchronous Time-Based Location Determination System". In: *Proceedings of MobiSys*. 2006. DOI: 10.1145/1134680.1134698.

[40] Zengbin Zhang et al. "SwordFight: Enabling a New Class of Phone-to-phone Action Games on Commodity Phones". In: *Proceedings of MobiSys*. 2012. DOI: 10.1145/2307636.2307638.