The Sound of Silence

Wai-Tian Tan*
Cisco Systems
dtan2@cisco.com

Mary Baker
Hewlett-Packard Laboratories
mary.baker@hp.com

Bowon Lee
Hewlett-Packard Laboratories
bowon.lee@hp.com

Ramin Samadani*
Qualcomm Technologies, Inc.
ramin_su@yahoo.com

ABSTRACT

A list of the dynamically changing group membership of a meeting supports a variety of meeting-related activities. Effortless content sharing might be the most important application, but we can also use it to provide business card information for attendees, feed information into calendar applications to simplify scheduling of follow-up meetings, populate the membership of collaborative editing applications, mailing lists, and social networks, and perform many other tasks.

We have developed a system that uses audio sensing to maintain meeting membership automatically. We choose audio since hearing the same conversation provides a human-centric notion of attending the same gathering. It takes into account walls and other sound barriers between otherwise closely situated people. It can sense participants attending remotely by teleconference. It does not require attendees to perform any explicit action when participants leave a meeting for which they should no longer have access to associated content. It works indoors and outdoors and does not require pre-populating databases with mapping information. For sensors, we require only the commonly available microphones on mobile devices.

Our system exploits a new technique for matching sensed patterns of relative audio silence, or *silence signatures*, from mobile devices (mobile phones, tablets, laptops) to determine device co-location. A signature based on simple silence patterns rather than a detailed audio characterization reveals less information about the content of potentially private conversations and is also more robustly compared across devices that are not clock synchronized. We evaluate our method in formal indoor meetings and teleconferences and in *ad hoc* gatherings outdoors and in a noisy cafeteria. Across all our tests so far, our approach determines audio co-location with a worst-case accuracy of 96%, and recovery from these errors takes only a few seconds. We also describe a content sharing application supported by silence signature matching, the limitations of our approach, current status, and future plans.

Categories and Subject Descriptors

C.5.3 [**Microcomputers**]: Portable devices; H.5.3 [**Group and Organization Interfaces**]: Computer-supported cooperative work; H.5.5 [**Information Interfaces and Presentation**]: Sound and music computing

General Terms

Experimentation, Measurement, Performance

Keywords

Mobile systems, mobile sensing, localization, audio, silence signatures, content sharing, mobile cloud services

1. INTRODUCTION

People meet with each other by necessity and by choice, formally and informally, in person and remotely. These different kinds of meetings can have very different characteristics. Sometimes there is a collaborative purpose that results in a shared artifact, a recorded consensus, or a group action. Sometimes there is no pre-planned purpose, but we may still value the discussion and would like to refer back to it and remember who was present. There are applications and tools to support many of these different kinds of meetings, including collaborative document editors, content sharing sites, teleconferencing systems, apps for exchanging business card information, photo sharing tools, and so forth.

Despite the wide variety of these applications, one underlying type of support seems generally useful: automatically knowing the group membership of the meeting. With a list of the dynamically changing group membership, we can enable easier content sharing with all the attendees. We can automatically exchange digital business card information and record the context in which we met the other participants. We can feed group membership information into calendar applications to simplify scheduling of follow-up meetings. We can automatically populate the membership of collaborative editing applications, mailing lists, and social networks. Half-way through a meeting, we can call another person who can then join us seamlessly, without painful interruptions for setup. With support from the infrastructure, we can effortlessly display content on ambient projectors and screens, as well as allow meeting rooms to determine when there are no longer people inside, enabling just-in-time conference room scheduling.

*Work performed while at HP Labs.

SenSys'13, November 11 - 15 2013, Roma, Italy

We believe that comparing audio sensed by different devices can help determine the group membership of a meeting whose attendees bring their mobile devices. Almost all mobile devices include audio sensors (microphones). Audio takes into account walls and other sound barriers between otherwise closely situated people. Our approach of comparing audio across devices gives us co-location information, rather than physical location, but it means we can use it anywhere without preparation – we do not need to pre-populate databases of mapping information. Hearing the same conversation provides a human-centric notion of attending the same gathering and can sense participants attending remotely by teleconference. It does not require attendees to perform any explicit action when leaving a highly sensitive meeting for which they should no longer have access to associated content.

We further believe that a new approach to audio sensing – comparing patterns of relative audio silence over time – brings some privacy and robustness advantages. Relative audio silence can be the space between voices, or it can be the space between sounds that are significantly louder than the ambient noise level. Currently, we capture these *silence signatures* on the mobile devices and stream them to a cloud service that matches them to form group membership lists that the service then sends back to subscribing applications. Because the cloud service receives only patterns of silence to compare, rather than more detailed features of the audio signals, we reveal the identities of attendees to the service, but not the content of their potentially sensitive conversations. While it is perhaps counter-intuitive that capturing less information is more useful, we find that silence signatures are easier to match than more detailed audio signatures, because they need not be closely time synchronized. They can also be more practical on some networks, since they require less bandwidth.

In the rest of this paper we describe the motivation for using silence signature matching and how it relates to other location-sensing techniques (Section 2). We describe the underlying algorithm(Section 3), and its current implementation and use in a mobile content-sharing application (Section 4). We evaluate our method (Section 5) in formal meetings in conference rooms and *ad hoc* gatherings outdoors and in a noisy cafeteria. We also explicitly test the method in difficult situations with phones in purses and pockets. Across all these tests we have so far observed a worst-case error rate of 4% which compares favorably with many other location-sensing technologies, and the system usually recovers in about two seconds from these errors. We also evaluate the power consumption of our implementation and its potential for scaling to large numbers of client devices, both of which remain problematic depending on how our technique is applied. Finally, we describe several known limitations of our method and our future plans (Section 6).

2. MOTIVATION AND RELATED WORK

Questions such as where we are, what we are near, and who is near us have long been important to people, so it is no surprise that the technology community continues to refine our location-sensing tools. In our current work, we are content to sense mobile computing devices in our proximity as proxies for the people who carry them, rather than the presence of the actual people. This presents us with a potentially easier problem than that solved by face recognition [41] and other technologies required by security surveillance and monitoring applications. Further, most of the applications we want to support already require people to carry their mobile devices with them. There are nonetheless many different approaches to sensing location and co-location using mobile devices. They have different strengths and weaknesses, and so far none is ideal in all situations and for all intents.

For our purposes, we are interested in answering the question "who are we with?" (co-location) rather than "exactly where are we?" (physical location). The bulk of the research in location sensing has gone into answering the latter question which is more inclusive: if devices could obtain accurate absolute locations and could map them accurately to the environment and compare results, then we could also determine who we are with.

Techniques to obtain sufficiently accurate location sensing, however, suffer from at least one of two drawbacks with respect to our goals. First, we want to sense co-location in as many contexts as possible (for instance both indoors and outdoors), while many location-sensing technologies work in only a subset of the contexts we need to consider. The Global Positioning System (GPS), for example, provides location information anywhere there is an unobstructed line of sight to four or more GPS satellites [29, 42], but it works with spread-spectrum signals of very low power (below noise floor) that are attenuated by building materials and thus does not generally function indoors.

Second, we want to sense co-location without the need for location-specific infrastructure or pre-populated mapping databases. Many absolute location-sensing technologies require support from local infrastructure or previously collected databases to match signals to locations. Indoor location sensing using WiFi [8, 10, 11, 16, 44] sometimes combined with Bluetooth [46] has become increasingly practical and accurate, with systems such as PinLoc [37] using PHY-layer fingerprints to localize users to a 1-meter-squared spot with almost 90% mean accuracy and fewer than 6% false positives. These solutions require a database that maps between location and network beacons or fingerprints. This location database may be previously calibrated, or it can be "crowdsourced" – developed over time by active users – but accuracy depends on its correctness and completeness. Similar solutions require matching other kinds of sensed information such as geo-magnetic fingerprints [14], images [32], acoustic background spectrum [40], and combinations of multiple kinds of fingerprints including images and the acoustic ambience of rooms [7]. One approach to avoid these databases is to apply dead-reckoning – sensing motion from compass, accelerometer, and gyroscope to chart a path from a previously known absolute location [15]. However, this approach does not yet seem to provide results accurate enough for us to determine co-location at the granularity of a small conference room. Some solutions use infrastructure such as active acoustic signals emitted by devices in the environment [35] or NFC tags [13] deployed in meeting rooms and other locations. Our aim is to sense co-location of devices regardless of whether we are indoors or outdoors, whether we have access to a particular kind of local infrastructure, or whether there is an accessible, well-calibrated database of sensed fingerprints against which to match our sensor readings.

Some systems, such as Virtual Compass [9], use peer-to-peer wireless communication to locate nearby devices and

thus answer the question "who are we near?" rather than giving us an absolute position. Because walls do not always significantly attenuate radio signals, these techniques may include nearby devices and not just devices that are part of the same conversational group or event. In Virtual Compass, devices maintain neighbor graphs through a two-phase algorithm in which they send messages to each other over both Bluetooth and WiFi and use the received signal strength of the messages to judge their respective distances. Because the messages contain neighbor lists and distances, nodes that are more than one hop away gain knowledge of each other. Virtual Compass has the advantage over signature-matching techniques of avoiding potential false positives, and by using multiple radios and peer-to-peer communication of neighbor lists it determines not just co-location but the relative placement of devices with respect to each other. Our technique does not determine relative placement, but it does not require direct communication between devices over multiple types of radios. Instead, we require that each device have a microphone and at least some mechanism for reaching our cloud service. Virtual Compass devices also use a cloud service that they access via GSM to help cut down on continual radio scanning to maintain their neighbor graphs. The Virtual Compass cloud service uses rough GSM localization to help determine when a new device might have entered the area and more scanning is required.

We believe that comparing sensed sound across devices, rather than radio communications, is a promising method for determining a human-centric notion of co-location. Sound respects meeting room walls and other barriers designed to keep separate groups of people from interfering with each other. Almost all mobile devices include microphones. Sound sensing is available almost everywhere and does not require sensing support in the local infrastructure (although we can happily make use of it, as we describe later in this paper). Another advantage of comparing sound across devices is that we can sense the virtual co-location of participants attending an event remotely through a teleconference, because all the participants hear the same conversations (as long as they are not on "mute" – a problem we also consider later in this paper).

Neary [30] is the system closest to ours that attempts to answer the same question "who are we with?" using sensed audio. Like our system, it is based on the idea that you are *with* the people who hear the same thing you hear. Participants use PCs equipped with microphones and networks. Every six seconds each device fills its six-second sound buffer, scans that buffer for the longest period of sound above a threshold, processes the sound with a Fast Fourier Transform, time stamps it, and then sends the results to all Neary devices. Receiving devices get the timestamp and user names from the received data, process the sound data with a Fast Fourier Transform and calculate the similarity of that data to the data the receiving device recorded at the same time. If the value is above a threshold, the devices are considered co-located. There are two issues with the Neary approach. First, because Neary uses Fast Fourier Transforms to characterize the sound bites, the recorded signals must be very well time-synchronized, or they will not compare. This is why Neary requires the devices to execute a peer-to-peer Network Time Protocol (NTP) for synchronization, which we do not require. Second, the characteristics of a sound signal heard on one microphone can differ substantially from the same signal heard on another microphone. This can be due to differences in the microphones or in the proximity of the microphone to the sound source. This manifests itself in Neary when two users talk at the same time. Because each voice is loudest at its nearest microphone, the two signals do not compare well.

Work by Zhang and Trott [45] employs the same general approach as Neary and exhibits the same issues because it too uses relatively complex acoustic signatures that attempt to represent rich information about the shape of the sound waves. This work also divides sound signals into temporal windows and computes the discrete Fourier transform of the windowed signal. To further reduce the size of the acoustic signature, the technique discards amplitude information in each frequency band and quantizes the phase information into two bits and sends these signatures to a central service for matching. This approach performs well when tested with synchronized captured data, but in practice the windows used to compute phase are not temporally aligned across devices, since device audio capture is rarely so precisely clock synchronized. This means the technique does not work well when confronted with network jitter, capture and transmission delays, and unsynchronized devices.

Instead of comparing sound signatures, we compute *silence signatures*. Our evaluations with a live system indicate that the use of silence signatures is robust to differences in loudness of the different devices and their placement relative to speakers. This means participants can talk over each other and also that the technique is not sensitive to moderate temporal misalignment of signals between devices. Compared to sound signatures, our silence signatures are also one seventh the size of those used by Zhang and Trott, and they provide more privacy with regards to conversational content. Wyatt et al. [43] also concern themselves with protecting the privacy of conversational content in their technique for separating speakers and their speaking turns in a multi-person conversation by "using only features from which intelligible speech cannot be reconstructed." We use a subset of just one feature and believe this makes it harder for third parties to reconstruct private audio content from silence signatures. (Our technique does, however, transmit device owner identities, since that is its purpose.) In work related to silence signatures, we have also performed a simulation study that focuses on the development of statistical models for the computation of a likelihood ratio to support sequential hypothesis testing to determine a variable-length window size for the signatures [39].

Other systems use sound in a variety of intriguing ways. BeepBeep [34] is a highly accurate range finding and localization technique that works indoors and outdoors and uses a clever mechanism to deal with uncertain latencies. For range finding it is similar to our system in that it requires no previous infrastructure. For localization, however, it requires anchors in known places against which it applies its Multiple BeepBeep Ranging scheme. SoundSense [27] scalably models sound events detected by mobile phones and uses supervised and unsupervised learning techniques to classify the sound types and discover sound events that are specific to individual users. It can then recognize these sound events in the daily lives of its users. Some systems use sound as an *over-the-air* modem for communicating content between devices [25, 6]. These systems have the advantage compared to

our approach that they can share device-resident media even if there are no other data networks available. Our approach currently requires at least some data network to be available for each of the mobile devices so they can communicate with a service in the cloud (although the choice of networks can differ across devices). Another sound-emitting method [19] uses sentences automatically generated from public keys and *vocalized* by a text-to-speech system to establish secure pairing between devices with the aid of humans for manual authentication. In contrast to these techniques, our method uses the unmodified acoustic environment – we need only sense the environment passively. Finally, speaker identification [24, 26] uses voice models derived from training data to map from voices to associated identities and would seem to solve our problem except that it requires a database of all potential speakers and would not sense the presence of people who are quiet in meetings.

Beyond the technologies we describe above, there are more active physical and virtual methods of associating devices with each other [12]. These techniques require explicit gestures such as bumping devices together [1, 20], shaking the devices [28], or tagging in with a technology such as NFC or foursquare [2]. These techniques can be difficult to scale to large meetings and some of them require an active gesture of disassociation rather than dynamically sensing when people leave meetings.

Matching patterns of relative silence meets our requirements for sensing the changing membership of a meeting, indoors or outdoors, without pre-existing local infrastructure or mapping databases. In this paper we further explore the technique's effectiveness and practicality.

3. ALGORITHM DESIGN

In this section we describe the signal processing techniques we use to compare conversations and other sounds heard on different devices to judge their similarity. The top two graphs of Figure 1 show a portion of audio recorded by an Apple iPad and a Samsung Galaxy S3. Due to different relative distances between the devices and the people who are speaking, the relative amplitude of the audio signals can be very different. This suggests that direct methods such as simple correlation of audio signals are not reliable in determining similarity. The middle two plots of Figure 1 show the energy of the sound signals over 10 ms intervals on a logarithmic scale. We can compute a silence threshold (described later and shown as a dashed line in these two plots) to quantize the signals into silence and non-silence. By removing natural but spurious silences of duration less than 100 ms, we obtain the *silence signatures* of the bottom two plots of Figure 1, where we indicate relative silence with a 1 and non-silence with a 0. Visually, the two signatures appear very similar, which is our goal. In the rest of this section, we describe how we classify silence, how we compute similarity between silence signatures, and how we choose various design parameters.

3.1 Acoustic Activity Detection

To form our silence signatures, we want to extract the silence that occurs in meetings. This task is related to Voice Activity Detection (VAD) except that we do not restrict ourselves to voice but consider general sounds. We call this Acoustic Activity Detection (AAD). In this section we describe how we prepare the signals for AAD and our choice

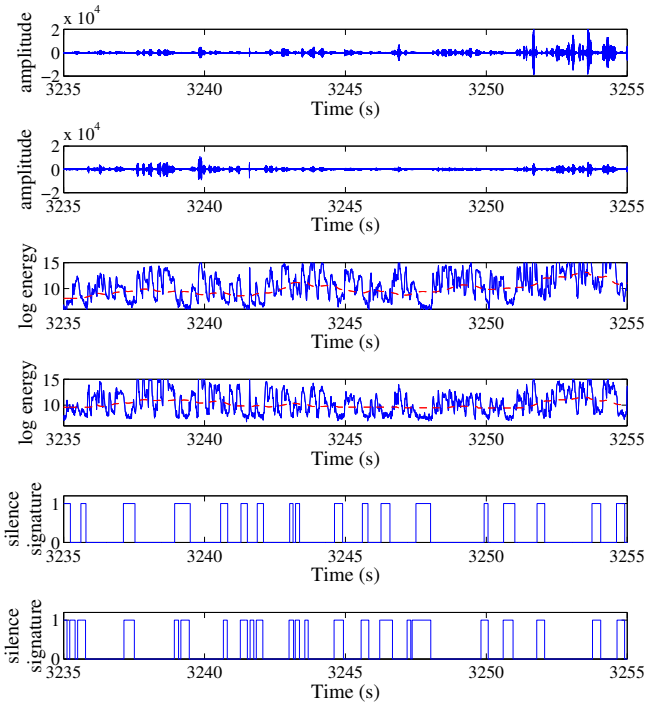


Figure 1: Sound signals versus silence signatures. The top two graphs show audio recordings from the same 20 seconds of a meeting on a Galaxy S3 (top) and an iPad (bottom). The middle two recordings show the energy of the sound over 10 ms intervals on a logarithmic scale. The dashed line is the silence threshold we compute. The bottom two plots show the silence signatures for these two recordings: 1 means silence and 0 means non-silence.

of acoustic activity detector.

In our experiments with various laptops, tablets and cell phones, we find some devices have low frequency electrical noise in the audio channel that, while not audible, deteriorates the AAD performance. To compensate for this, we filter the audio signals with a high pass filter to cut out this potential low frequency noise. We designed a 51 tap Kaiser window Finite Impulse Response filter [29] with a stop band from 0 to 150 Hz, a pass band from 400 to 4000 Hz, and a stop band attenuation of 30 dB. We sample audio signals at 8000 Hz. The top two plots of Figure 2 show the 15-second audio signal of a meeting captured by two different devices. The bottom plot shows the filtered output of the noisy signal in the middle. We find this improves AAD performance across all the devices.

We determine co-location based on similarity of silence patterns across devices. This means our choice of silence signature can affect our results. As shown in Section 5.2, we can make use of VAD signatures, but we prefer to use AAD. AAD is better for our purposes because it works across more environments by exploiting non-voice as well as voice cues, and because it is simpler. There are many existing VAD algorithms, but they are designed for other applications. For example, many low-rate speech compression engines code and transmit data only when VAD determines that there is speech content [23]. VAD designed for speech codecs tends to misclassify silence as speech. This is understandable for

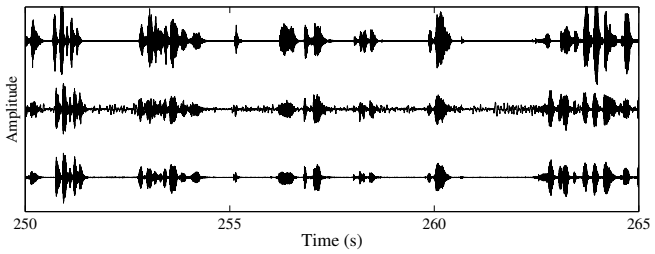


Figure 2: Filtering out low-frequency noise. The top two plots show 15-second audio signals captured by two devices. The bottom plot shows the results of the noisy signal in the middle after filtering.

speech compression, since misclassifying speech as silence causes drop-out that reduces intelligibility, while misclassifying silence as speech only corresponds to a minor increase in bit-rate. VAD uses a variety of features in its algorithm whereas we use only the energy level of the signal. Our simpler method appears to be more robust in some situations as described in Section 5.2 where we further compare our AAD against a popular VAD algorithm.

Generally VAD attempts to separate recorded audio into segments corresponding to ambient noise and speech. The algorithms usually learn ambient noise characteristics from the signal itself over a certain period of time. For our purposes, we find that it is not ideal to use a constant threshold to determine silence even for a few seconds. Specifically, if we refer to the middle plots of Figure 1, we see that both the "lows" and "highs" have significant variation over time, making a continuously changing threshold more appropriate. For AAD, we thus generate the silence threshold (dashed line) in Figure 1 by starting with the log-energy sequence $\mathbf{e}$ computed over 10 ms frames. We first compute a smoothed version $\tilde{\mathbf{e}}$ using the moving average of 9 consecutive frames [31]. For every sample $\mathbf{e}_i$ at each frame, we compute the minimum value m_i and the average value a_i of $\tilde{\mathbf{e}}$ over a 2-second window W_i centered at the sample. We then compute the local threshold τ_i as an intermediate value between the local minimum and average.

$$\tau_i = m_i + \max\{2, 0.7 \times (a_i - m_i)\} \quad (1)$$

We expect in most useful conversations that a speaker will speak loud enough over the ambient noise to reach the intended audience, so we require the sound threshold to be at least 2 dB over the local minimum. Note that this assumes that over a 2-second interval, there is 10 ms of signal that is close to the ambient noise level even in continuous speech, which is generally the case given natural pauses even in the middle of sentences.

3.2 Signature Matching Metric

Here we describe our approach to matching silence signatures to determine co-location. Given silence signatures $\mathbf{s}_0$ and $\mathbf{s}_1$, we need a similarity measure that gives a value close to 1 for co-located $\mathbf{s}_0$ and $\mathbf{s}_1$ and a value close to 0 for unrelated $\mathbf{s}_0$ and $\mathbf{s}_1$, so that we can readily determine similarity by threshold. We can then use these similarity measures independently each time, or we can use thresholds dependent on past decisions. In this paper, we adopt the simpler approach in which decision thresholds remain constant.

Parameter	Value
Silence signature length	20 seconds
Interval to compute energy	10 milliseconds
Interval to characterize ambience	2 seconds
Significant sound threshold	2 dB
Signature upload frequency	every 2 seconds
Server timeshift range	500 milliseconds

Table 1: Values for various parameters used in this paper.

We use a similarity metric based on the cosine metric

$$C(\mathbf{s}_0, \mathbf{s}_1) = \frac{(\mathbf{s}_0 - \bar{s}_0)}{|\mathbf{s}_0 - \bar{s}_0|} \cdot \frac{(\mathbf{s}_1 - \bar{s}_1)}{|\mathbf{s}_1 - \bar{s}_1|} \quad (2)$$

where we treat $\mathbf{s}_0$ and $\mathbf{s}_1$ as vectors with scalar means $\bar{s}_0$ and $\bar{s}_1$, respectively, and the dot represents the vector dot-product. When $\mathbf{s}_0$ and $\mathbf{s}_1$ are unrelated, $C(\mathbf{s}_0, \mathbf{s}_1)$ is close to zero, but when $\mathbf{s}_0$ and $\mathbf{s}_1$ are from the same time and place, $C(\mathbf{s}_0, \mathbf{s}_1)$ is close to one. Geometrically, C is the cosine of the angle between vectors $\mathbf{s}_0 - \bar{s}_0$ and $\mathbf{s}_1 - \bar{s}_1$ and has a range of [-1 1].

3.3 System Parameters

Systems design often involves choosing design parameters. Here we explain our choices for the more important parameters. Table 1 lists their values.

First, we compare silence signatures of length 20 seconds to determine co-location. A longer signature gives higher accuracy when there are no changes in co-location status, but it also increases decision latency when co-location changes occur. Generally, for dynamic audio content with significant alternations between loud and soft sound, a shorter signature may suffice, but we find that 20 seconds is better for common use. In the future we may further explore an adaptive signature length based on the nature of the audio content.

Next, we determine silence or sound values for 10 millisecond intervals. Ten milliseconds is a common choice in audio processing, since one can assume that the signal remains largely stationary within each interval, and the intervals are small enough to provide fine granularity for robust matching. In Equation 1, we determine the minimum and average energies using a 2 second window. Empirically, we have found this choice sufficient to respond to changes in microphone gain control in common mobile devices. Similarly, in Equation 1, the decision threshold needs to be at least 2 dB above the corresponding minimal energy to avoid instabilities when the average energy is close to the minimum energy, e.g., in a relatively noise-free room.

As we describe in the next section, a device transmits every 2 seconds its newly generated silence signature to a server for comparison. More frequent updates improve responsiveness at the expense of higher network overhead associated with transmitting more messages.

Finally, to address differences in network delay and jitter, the signature matching server performs a timeshift of up to +/- 500 milliseconds when comparing signatures from two devices. This value is large enough to accommodate large-delay networks such as 4G. Disadvantages of using a large value are increased computation and a larger chance of incurring false positive matches.

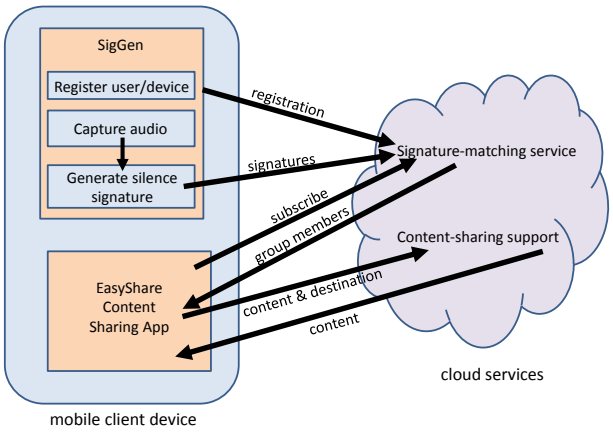


Figure 3: A diagram of the content-sharing application. The SigGen silence signature generator runs on the mobile devices sending silence signatures to the cloud for determining co-location. The cloud service sends a list of co-located devices to all the devices in the group. This information is used by applications such as the EasyShare content-sharing application that also uses a cloud service to share content between devices.

4. SYSTEM AND APPLICATION

In this section we describe our implementation of the silence signature co-location service and its use in a very simple content sharing application. Figure 3 illustrates our implementation. The mobile clients perform audio capture and generate silence signatures and also provide a user interface (UI) that displays the current group of co-located devices and a way to browse and share content with the other members of the group. Ideally we would implement the client components entirely in HTML5 since we are aiming for a cross-platform solution. However, some mobile browsers do not yet support audio capture without also capturing video. As a result, we temporarily adopt a split approach and have implemented a native audio capture and signature generating application for Android and Windows 8 and a content sharing UI in HTML5, both of which use websockets to communicate with a server in the cloud. This enables the clients to function behind firewalls, at home, at work, and elsewhere.

The audio capture and signature-generating application, called SigGen, captures 16-bit mono audio at 8000 Hz, and computes a binary silence value every 10 milliseconds. This corresponds to an unencoded rate of 100 bits per second. Every 2 seconds, SigGen sends the 200-bit or 25-byte silence signature to a server. The server compares the signatures and sends back a list of group members to applications that have registered with it (in this case the content-sharing app).

The sharing UI, shown in Figure 4, displays icons (currently just user names and device-type abbreviations) for members of the co-located group. These icons come and go as users join and leave the group. There are many ways to gather the user information: enterprise directories, LDAP servers, and online authenticated identity servers from various sites and vendors. The appropriate choice of identity information depends on the application developers' purpose. The icons are draggable so that users can arrange them on the screen to represent the actual positions of physically co-located attendees if that is useful to them.

Below the user icons is a content browsing space where users can put content to share and where shared content appears. Content that has not yet been shared has green borders, while shared content (both sent and received) has red borders. To share content the user just drags it and drops it in the user icon area (to share with the entire group) or onto individual icons (to share with individual users).

HP's cloud service [3] hosts both our signature co-location service and support for our content-sharing application. The signature co-location service (which is very parallelizable) receives silence signatures from each device every two seconds, and keeps past signatures for a couple of minutes to support matching with larger time windows. The server timestamps the received data using its own clock and compares the signatures pair-wise using the latest 20 seconds available from the two devices. Network jitter can cause inaccuracies in recorded time, so we time-shift computed silence signatures by +/- 50 blocks (500ms) to account for the jitter. Currently it takes about 10 seconds to drop a device from a group. This helps smooth out audio drops, jitter, and longer-than usual silences in conversations.

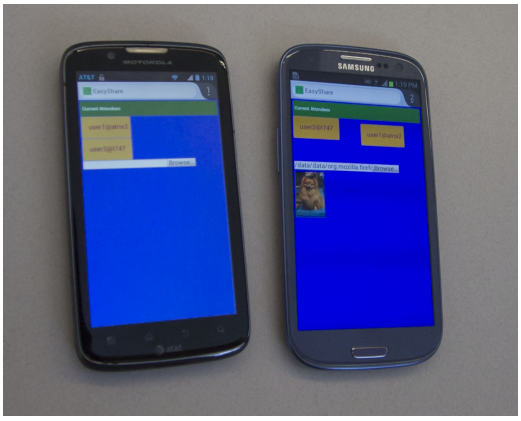
5. EVALUATION

We evaluate our solution in several ways. We first look at the accuracy of our signature-matching technique using our AAD. We do this by taking silence signatures from many hours of actual meetings and using them as input to our matching algorithm to see if this generates any false positives or false negatives. We then compare the results of the matching algorithm on the same recordings using signatures generated instead by a conventional G.729B voice activity detector [22] that is popularly used for speech coding for mobile telephony. We next look at the accuracy and behavior of our overall system in a variety of challenging situations, including a teleconference, scenarios when device microphones are muffled by being stored in pockets and purses, and over a network with significant jitter. We then examine the amount of time it takes to react to a change in the co-location status of a group member. We next measure the power consumed by our implementation of silence signature generation (the SigGen app) on mobile phones. Finally, we explore the potential for our cloud-hosted co-location sensing service to scale up to large numbers of devices.

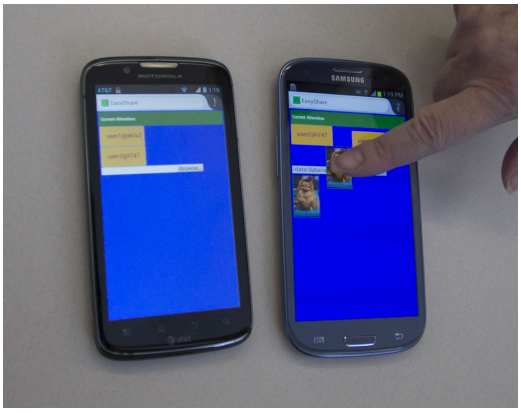
5.1 Accuracy

We evaluate the accuracy of our silence signature matching technique in isolation from the rest of the system by comparing the technique's results against ground truth using data collected by carrying our devices to eleven meetings, each with 3 to 7 people. Using silence signatures generated from these meetings we look for both false negatives (we fail to show as co-located signatures from the same time and place) and false positives (we show as co-located signatures that are not from the same time and place). We summarize the settings and other parameters of the meetings in Table 2. For each device at each meeting we generate a 20-second silence signature for each 30-second segment of audio.

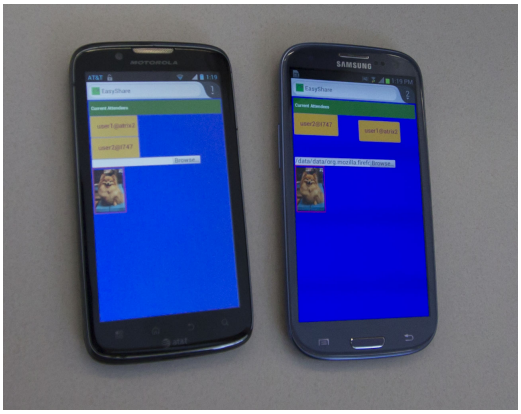
Using these signatures, we first test to see if we correctly identify all co-located segments. We do this by testing for each participant the similarity of each 30-second segment



(a) Two phones are detected as co-located. On the right phone a user has selected a photo to share.



(b) The user drags the photo to share.



(c) The shared photo appears on both devices.

Figure 4: Content sharing application. The top half of the display shows names of users in the co-located group. The space below shows content for sharing, and we show a user sharing a photo.

pairwise with the same segment recorded by each other participant. We do this for every time segment except the first and last, which gives us 4998 pairs of silence signatures that we know should be sensed as matching.

We then test to find if we falsely match any segments that should not be considered matching. To generate the same number of pairs of signatures that should not match, we use the same co-located pairs, but we compare signatures shifted by one 30-second segment. Note that because our "unrelated" pairs are time-offset versions of the same meeting, they are more likely to have similar statistics and are in general harder to distinguish than recordings from different settings, making this a challenging test. (Table 2 also shows the number of pairs contributed by each meeting. For example, the first 285 pairs in Figure 5 are from meeting 1, the next 6 pairs are from meeting 2, and so on.)

id	no. of people	length (s)	no. of pairs	description
1	3	2884	285	lunch in noisy cafeteria
2	3	100	6	teleconference 1
3	3	385	33	ad hoc discussion
4	5	1700	550	brainstorming: room 1
5	4	1210	234	brainstorming: room 2
6	4	2500	492	brainstorming: room 3
7	7	3488	2415	project meeting 1
8	4	3440	678	project meeting 2
9	3	1364	132	lunch outdoors
10	2	2970	98	teleconference 2
11	2	2280	75	4G & WiFi in cafeteria

Table 2: Duration and number of co-located recordings in various meetings. The column for number of pairs is explained in the text.

Figure 5 shows the results for our cosine similarity measure. Using our cosine measure, unrelated pairs should have a score falling closer to 0, while matching pairs should have scores closer to 1. We can see from the top plot in Figure 5 that this is indeed true: a decision threshold of 0.4 successfully classifies all true matches (shown in black) as matching, except for one pair, and all unrelated pairs shown in gray as non-matching, except for two pairs. This gives us a 99.97% accuracy for our signature matching technique for the data from this set of meetings.

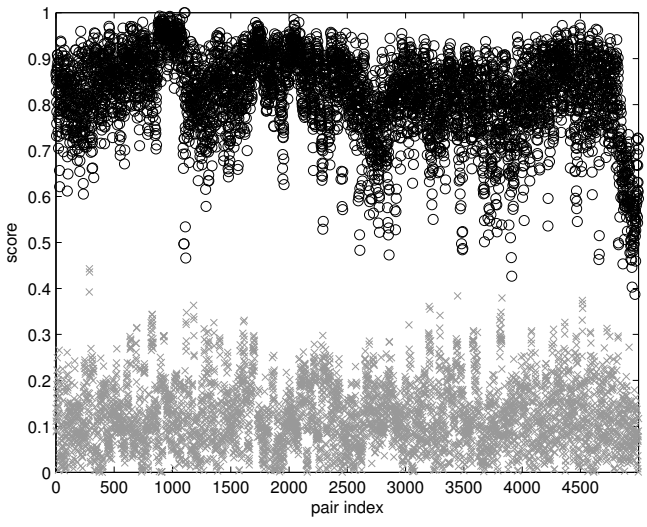


Figure 5: Score for 4998 pairs of co-located (black) and 4998 pairs of non-matching (gray) silence signatures using our cosine similarity measure.

The noisy cafeteria lunch scenarios are noteworthy because the false positives and negatives occurred during these two meetings, giving only a 96% signature matching accuracy for that subset of data. We expected worse results, since if an environment is noisy enough that there is no relative silence, the signatures will not carry useful information. However, the service worked well in practice because people speak up to be heard over the ambient noise, and so their voices are usually sufficiently above threshold to generate a useful signature. We also find that because people tend to adjust their voices to be heard by their intended audience only, a potential eavesdropper running our application at the table next to ours was appropriately not sensed as co-located. Nonetheless, sound can travel in odd ways, bouncing off of surfaces and making it hard for people to determine visually whether they will be heard by others. To address this, we believe it will be helpful in these situations if applications have a way to pick and choose which peripherally sensed people to consider co-located.

5.2 Comparison of AAD and VAD

How useful are our AAD silence signatures in comparison with signatures generated by a typical voice activity detector? We chose AAD because it is simple and works with general acoustic signals and not just voice, but it needs to perform at least as well as VAD in meetings and conversations to be a good choice. In this section we provide some comparisons.

Figure 6 shows plots of signatures from audio recordings made in the same meetings in Table 2 using our AAD algorithm and the conventional G.729B VAD [22] algorithm using a "hang-over" scheme that smooths out some voice/non-voice transitions. For consistency, we flipped the usual output of VAD to show instead a 0 for voice and a 1 for non-voice to generate comparable silence signatures, except that "silence" means no voice detected, while "non-silence" means voice activity. We see that VAD generates many more transitions than AAD, despite the use of the hang-over scheme. Consequently, the signatures generated by AAD in the top two plots are more clearly similar than those of VAD in the bottom two plots.

To compare AAD and VAD scores, we repeated the same set of experiments in Section 5.1 with signatures generated by the G.729B VAD [22]. We used the first nine meetings of the recordings in Table 2 because we did not have raw audio recordings saved for the last two meetings. For meetings 7 and 8 an HTC EVO 4G device joined the group. On that device the G.729B VAD had trouble generating good signatures, due to a non-linear affect on some of the features used by VAD perhaps caused by the device's implementation of automatic gain control. Our AAD worked fine on this device, but VAD detected 100% and 99% speech for parts of meetings 7 and 8 respectively, where the correct speech percentage was around 70%. We excluded the recordings from this device since they tripped up VAD algorithm, making the total number of pairs 3796 instead of 4998.

Figure 7 compares the results, where Figure 7(a) shows the results generated with the silence signatures by our AAD. A decision threshold of 0.45 successfully classifies all true matches as matching (except for one false negative pair), and all unrelated pairs shown in gray as non-matching (no false positives). This gives a 99.9% accuracy on this data set for our signature matching technique with AAD. Figure 7(b)

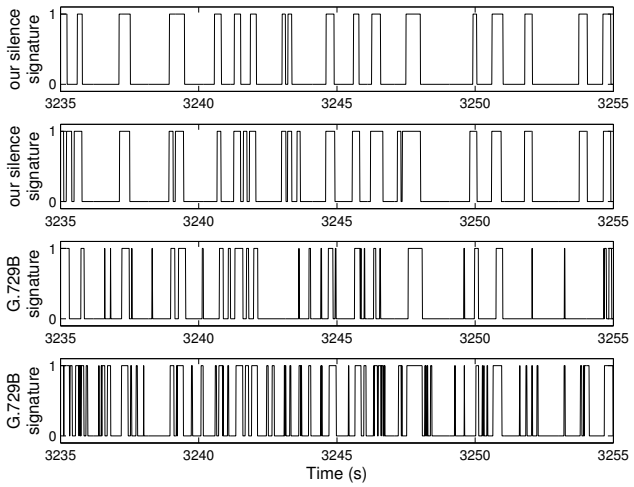


Figure 6: Comparison of different signatures. The top two graphs show the silence signatures generated by our AAD algorithm for the recordings in Figure 1. The bottom two graphs show those generated by the G.729B VAD [22] algorithm.

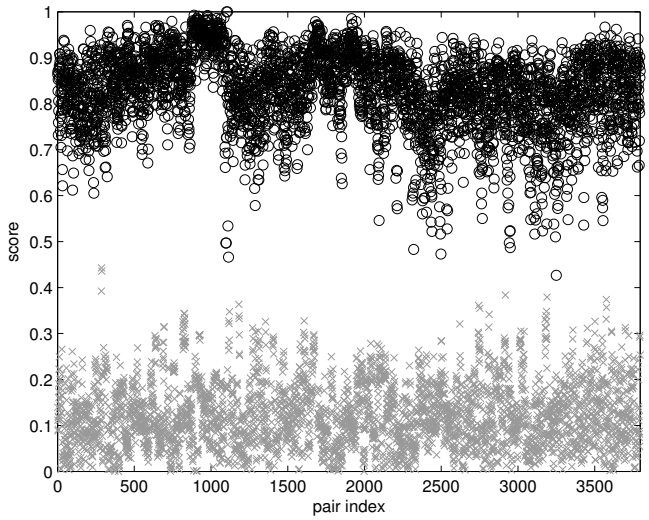
shows the results using the silence signatures generated by the G.729B VAD. A decision threshold of 0.33 gives a minimum overall error of 17 pairs consisting of five false positives and twelve false negatives. This is an accuracy of 99.8% for this data set. While there are other VAD algorithms to explore, we believe these results indicate that silence signatures are a robust signal for comparison whether they track acoustic or voice silence, that AAD is at least as viable for our purposes as VAD, and that by considering only the energy level of the signal, the simplicity of AAD may allow it to generate usable signatures in circumstances where VAD has more trouble. (Accuracy for the VAD algorithm including the bad signatures was 89.2%.)

5.3 System Accuracy in Difficult Scenarios

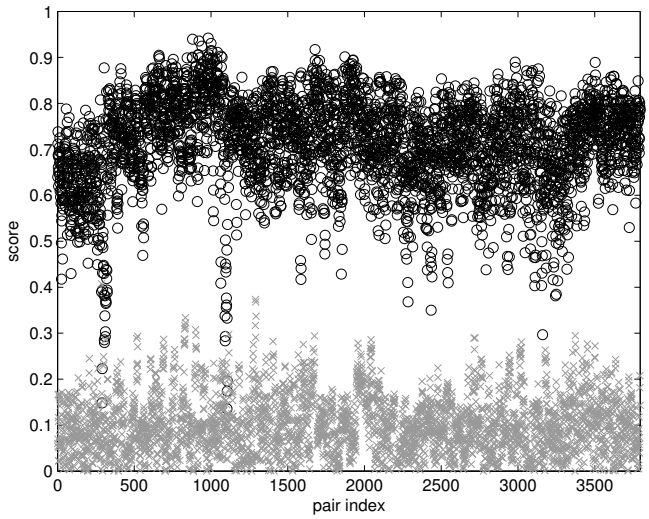
We next explore the accuracy and behavior of our overall system in three particularly difficult situations. For these results we use the co-location decisions logged by our signature matching service. These results thus reflect any overall system problems, including audio drops, delays sending signatures, network jitter, and so forth.

5.3.1 Teleconferences

One of the novel advantages of our technique is its ability to sense as co-located users attending meetings over the phone. Audio signals locally and over the phone are quite different. This is why voices over the phone sound different to the human ear from voices in the same room. For the same reason, many approaches that compare a rich characterization of the audio signals will not work well for our purposes. In contrast, comparing only extracted patterns of silence accomplishes our goal. Figure 8 shows the co-location decisions made by our live system during a 25-minute teleconference for each of two local users in different parts of a conference room with respect to a remote user attending over the phone. For this test, the remote participant joined the conference via a commodity speaker phone (on both ends of his connection), and all participants ran our SigGen service on their mobile phones, which they kept near them on



(a) Score with our AAD.



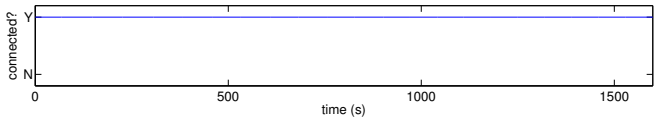
(b) Score with G.729B VAD.

Figure 7: Score for 3796 pairs of co-located (black) and 3796 pairs of non-matching (gray) silence signatures using our cosine similarity measure.

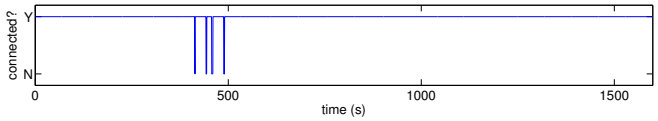
top of the desk or table at which they were sitting. For this teleconference we see a worst-case accuracy of 99.5%. (Co-location accuracy was 100% between the two local users in the same conference room.)

5.3.2 Purses, Pockets, and Pants

Mobile devices are sometimes stuffed into purses and pockets that may muffle their sound environment so that the devices do not capture a representative silence signature. In enterprise or formal group meetings where many people bring laptops and tablets to the table, this may be less often a problem than it is for *ad hoc* gatherings, say in front of a Peet's coffee shop. Using our live system, we explicitly test the accuracy of our technique in three of these potentially difficult situations. In the first scenario, one of the users keeps his phone in his shirt pocket with the microphone facing his chest so that it is most likely to be muffled



(a) remote user and local user 1 (100% accuracy).



(b) remote user and local user 2 (99.5% accuracy).

Figure 8: Co-location results for a teleconference over a Polycom.

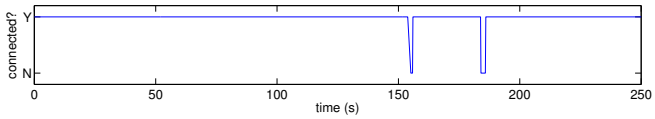
by his body. In the second scenario, one of the users keeps his phone in his front pants pocket, with the microphone turned outside facing the arm of the chair he is sitting in, which squeaks. In the last scenario, one of the users keeps her phone in a zipped-up purse, alongside her keys, but the purse is near her during the meeting.

The most difficult situation for our service among these tests is when the phone was in a user's pants pocket. In this particular situation, the phone was pressed up against the arm of the chair which seems to have acted as an acoustic reflector. The chair also squeaked as the user fidgeted, which created enough of a separate acoustic environment that we achieved only a 97.9% co-location accuracy. In practice, this level of accuracy still allowed our application to work well, although we did see the user's icon flicker on the screen a few times. The second hardest situation from these tests is keeping the phone in a user's shirt pocket, facing his body. This not only muffled the microphone but picked up the user's heartbeat. Accuracy in this case was 98.4%. We were surprised to find that putting the phone in a user's zipped purse had so little effect on accuracy (99.9%), since the purse was quite thick.

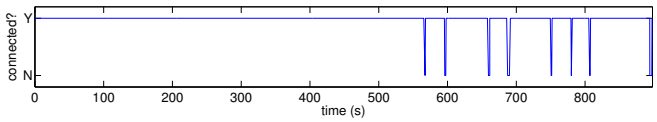
These tests by no means cover all the possible ways to muffle device microphones, and if the user sufficiently separates the device's microphone from his own acoustic environment, then our system will not work. Nonetheless, these tests suggest that our technique can be practical in at least some of the situations that occur if users put away their mobile devices when they are not actively interacting with an application that makes use of our service.

5.3.3 Dealing with Network Delay and Jitter

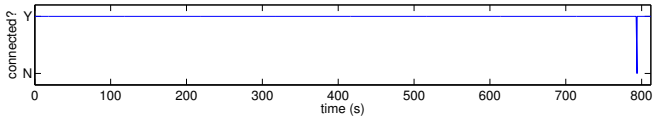
We can only usefully compare two silence signatures for co-location if they are captured at the same time. Mobile devices stream the silence signatures to our server, and there are many possible sources of delay and timing variation between the device capturing the sound, generating a signature, and that signature's reception at the server. In our experience, the most noticeable source of delay is network jitter. One of the advantages of a silence signature is that it consists of a one-bit value (silence versus sound) for every 10ms (in our implementation). This means it is easy on the server to timeshift the signatures by 10ms increments until we have compensated for the jitter. Figure 10 shows how changing the allowable timeshift from 250ms to 500ms improves accuracy for meetings where at least one of the devices experiences significant jitter. For these results we used traces at our server, including the logged signatures,



(a) inside shirt pocket

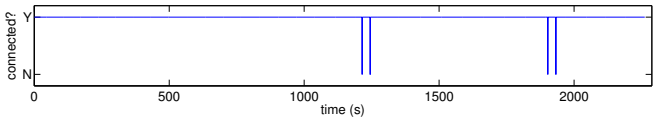


(b) inside pants pocket

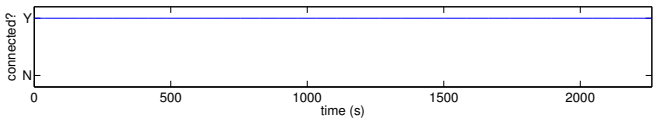


(c) inside purse

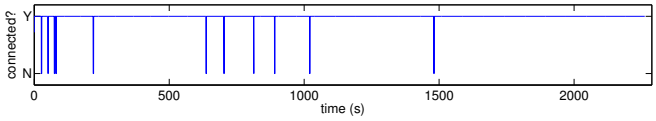
Figure 9: Co-location results for three difficult situations. We recorded a 97.9% accuracy with the phone in a pants pocket, a 98.4% accuracy with the phone in a shirt pocket, and a 99.9% accuracy with the phone in a purse. Note that these tests ran for different lengths of time.



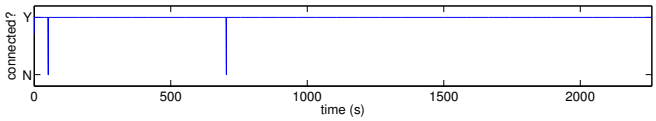
(a) 250ms timeshift, WiFi network (99.7% accuracy)



(b) 500ms timeshift, WiFi network (100% accuracy)



(c) 250ms timeshift, 4G network (99.1% accuracy)



(d) 500ms timeshift, 4G network (99.8% accuracy)

Figure 10: These graphs compare signature matching accuracy for different server timeshift ranges. In practice, we use 500ms because it is large enough to accommodate most jitter on a wide-area network, as seen here. With only 250ms we see errors even on a WiFi network (top two graphs), and more pronounced errors on a 4G network (bottom two graphs).

and replayed the traces with a 250ms available timeshift to compare it to our usual 500ms timeshift. In this figure, the devices all participated in a meeting in a noisy cafeteria, but two of the devices used a relatively jitter-free WiFi network while one of the devices used a 4G network. At the server, 3.2% of the signatures from the device using 4G experience jitter larger than 100 ms, while 2.1% and 0.1% experience jitter larger than 250 ms and 500 ms, respectively. The top two graphs show that matching between the WiFi-using devices improves from 99.7% accuracy to 100% accuracy by using the longer comparison window. The bottom two graphs show that matching between a WiFi-using device and the 4G-using device improves from 99.1% accuracy to 99.8% accuracy with the longer comparison window. It is still possible for large network jitter to cause matching failure in true co-located signatures, but recovery occurs whenever the next signature (sent every 2 seconds) arrives within the temporal search range of the server. Thus recovery usually occurs in about 2 seconds.

5.4 Co-location Sensing Latency

We next look at the latency for co-location decisions. Figure 11 shows how the cosine similarity score changes between meeting participants as one leaves the room. User 2 and User 3 are in one conference room having a teleconference with User 1 who is attending remotely by phone. User 2 leaves the room at time 105 seconds and returns at time 310 seconds. Our similarity score correctly shows that User 1 and User 2 (the top graph) remain co-located. The bottom graph, which plots the similarity score between User 2 and User 3, indicates that User 2 leaves the room at time 114, which is a 9-second decision latency. The similarity measure senses that User 2 returns to the room at 320 seconds, which is a 10-second decision latency. Across our tests, we find the average latency to sense co-location changes is about 10 seconds, or one-half the length of a silence signature. There is a trade-off between signature matching accuracy and the time to detect co-location changes: increasing the length of signatures will decrease the chance of false positives for our signature matching service, but it will also increase the time it takes to detect co-location changes.

5.5 Power Consumption and CPU Load

Managing battery lifetime is always a concern on mobile devices, and any new sensing application that might run frequently is especially worrisome. We test the power consumption of SigGen – our client-side software that performs audio capture, generates silence signatures, and sends them to the server. SigGen sends a 200-bit silence signature to our cloud service every two seconds. In this test, it does so over a WiFi connection. We test SigGen as a whole in what we believe is its worst-case scenario – running continuously as a service in background rather than being triggered as necessary by an application. Using a Monsoon power monitor we recorded SigGen consuming an average power of 199mW on a Samsung Google Nexus S running Android 4.1.2, and 191mW on a Samsung Galaxy S4 running Android 4.2.2, including its use of WiFi, CPU, and audio recording pipeline. We derived each average from 15 minutes each of readings with and without the service running.

The CPU load of SigGen running continuously averages 4% on the Samsung Galaxy S3, 3% on the Samsung Google Galaxy Nexus S, 7% on the Samsung Galaxy S4, and 8% on the LG Optimus Elite, which is the cheapest Android phone

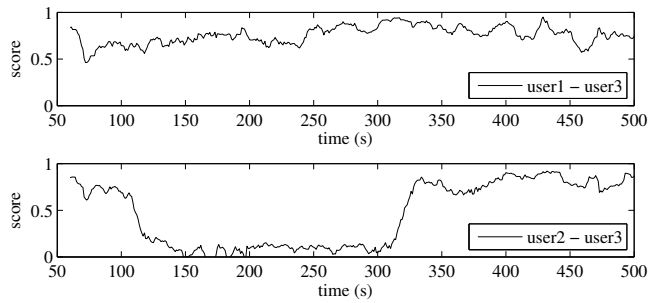


Figure 11: Changes in similarity score between User 1 and User 3 (top) and User 2 and User 3 (bottom) as User 2 leaves and re-enters the room.

we could purchase and which has a single core running at 800MHz (as reported by the Android app "OS Monitor" from *eolwral*). Most of the computation is in the direct implementation of high-pass filtering, which we apply to captured audio at 8000 Hz. Without the filter, our CPU load is 1% on the Samsung Galaxy S3. Our current filter implementation is not optimized; it uses a 51 tap Finite Impulse Response filter implemented with floating point numbers. In future work we may optimize this filter, but our experience is that on most newer mobile devices the filter is no longer necessary, and we may eliminate it entirely.

Power consumption remains a problem for our technique if it is run continuously rather than being triggered when needed by an application. For continuous use, the authors of Virtual Compass [9] face a similar problem and demonstrate the benefits of reducing WiFi activity, since it is particularly power hungry. In SigGen we can do this by batching signatures for less frequent transmission. The drawback is an increase in latency for detecting membership changes, which may be acceptable for many applications of the service. We also hope that phones that apply low-power CPUs to particular sensors, such as Motorola's Moto X, will prove useful in the future for understanding when to activate services such as ours.

5.6 Scaling Up to Many Devices

Aside from power consumption, one of the biggest concerns about our silence signature matching service is how large a set of devices it can manage. A global service might have hundreds of millions of customers. A service deployed inside a large enterprise might have hundreds of thousands. How silence signature search scales with the user population depends on the type of application, and some use models will scale much better than others. As numbers grow, we believe it will be necessary to apply a variety of means for reducing the search space, to reduce both search time and the potential for false positives.

Exploiting an application's particular needs or extra context is likely to be the most helpful method for reducing the search space. For instance, a content sharing application manually invoked by users to share among people who are *physically* co-located scales very well. Manual invocation needs only a low cloud computing duty cycle, with the resources needed only at the relevant time and only by those triggering invocation in that time frame. In addition, limiting groups to physically co-located people means that we can make use of other information such as past GPS readings taken while the device was outdoors or device IP addresses to prune the search space.

On the other hand, large-scale applications that require both dynamic monitoring of group membership and accommodation of remote group members joining from unpredictable places have the most intense needs. Dynamic monitoring requires matching a larger set of silence signatures at any particular time. The accommodation of remote group members, especially those who can join from arbitrary locations, means that other signal information such as a past GPS reading or IP addresses might not be useful to prune the search space.

Fortunately, the cosine similarity measure is easy to evaluate, especially for binary signatures. Furthermore, we believe that our problem is appropriate for applying known techniques for efficient scaling of search such as locality-sensitive hashing (LSH) [38] and multi-dimensional data structures [36]. In particular, LSH is capable of improving search even with high dimensional data. In our application, LSH is used to prune the set of candidates to be tested for co-location. We discuss fast computation of cosine similarity and using LSH for pruning in the next subsections.

5.6.1 Fast Computation of Cosine Similarity

Our cosine similarity measure may appear computationally expensive at first, but it mostly involves counting and addition operations that scale linearly with the number of bits in the signature.

Given N-bit binary silence signatures $\mathbf{s}_0$ and $\mathbf{s}_1$ with n_0 and n_1 number of ones, respectively, we have

$$\begin{aligned} |\mathbf{s}_0 - \bar{s_0}|^2 &= (\mathbf{s}_0 - \bar{s_0}) \cdot (\mathbf{s}_0 - \bar{s_0}) \\ &= \mathbf{s}_0 \cdot \mathbf{s}_0 + N\bar{s_0}^2 \\ &= n_0 + n_0^2/N \end{aligned} \tag{3}$$

and similarly,

$$\begin{aligned} (\mathbf{s}_0 - \bar{s_0}) \cdot (\mathbf{s}_1 - \bar{s_1}) &= \mathbf{s}_0 \cdot \mathbf{s}_1 + N\bar{s_0}\bar{s_1} \\ &= \mathbf{s}_0 \cdot \mathbf{s}_1 + n_0 n_1/N \end{aligned} \tag{4}$$

and combining, we can rewrite (2) as

$$C(\mathbf{s}_0, \mathbf{s}_1)^2 = \frac{(\mathbf{s}_0 \cdot \mathbf{s}_1 + n_0 n_1/N)^2}{(n_0 + n_0^2/N)(n_1 + n_1^2/N)} \tag{5}$$

where we can obtain n_0 and n_1 by simply counting the number of ones in $\mathbf{s}_0$ and $\mathbf{s}_1$ respectively. We can compute $\mathbf{s}_0 \cdot \mathbf{s}_1$ using binary-and followed by summing. Thus the computational cost of evaluating (2) is low.

In the server after evaluating $C(\mathbf{s}_0, \mathbf{s}_1)$, we typically shift $\mathbf{s}_1$ by one sample and re-evaluate against $\mathbf{s}_0$. In that case, we can realize further computational savings since n_0 remains the same, and n_1 can be adjusted without going through the new $\mathbf{s}_1$ to recount.

5.6.2 Using Location Sensitive Hashes

LSH techniques are useful for a variety of similarity measures [5, 17, 18, 33] but we focus on the cosine similarity measure that seems to perform well for our application (Section 3), and we provide preliminary evidence for improved scaling based on a simple LSH algorithm. A typical LSH proceedure for the cosine similarity measure starts by calculating hashes of the form $h(s) = sign(\mathbf{r}^T\mathbf{s})$, which output the sign of the projection of the silence signature $\mathbf{s}$ onto the random vector $\mathbf{r}$ with $\mathcal{N}(0, 1)$ independent, identically

distributed normal elements. To improve the search statistics, we concatenate the outputs of K independent random projections of the form $h(\mathbf{s})$ to form multiple new hashes $g_j(\mathbf{s})$ for $j \in [1, L]$. For efficient search, each hash $g_j(\mathbf{s})$ has pointers to all the silence signatures that hash into it. A subsequent verification phase for the signatures in the common hash tables computes the cosine, as described in Section 3, for all of the candidate signatures that are also pointed to by the hashes $g_j(\mathbf{s})$.

Two tests determine the performance of the LSH. The first test with a large set of co-located signatures determines the percentage of missed co-located candidates – co-located candidates that should fall into the same set of hashes but do not. This test shows a good miss error performance of about 1% while pruning the search space by about 63%. The second test with a large set of non-co-located signatures determines the efficiency of the pruning: how many non-co-located signatures fall into the same set of hashes when they should not. We conduct the tests by forming hashes with $K = 9$ and $L = 33$ from silence signatures of 20-second duration (corresponding to vectors with 2000 time samples). We use the same data as described in Section 5.1 to evaluate the hash. We process the 4998 pairs of co-located silence signatures from which there are 49 false misses. A miss in this case means that we fail to sense the co-location of a signature. For our second test we use 24690 non-co-located pairs of silence signatures and find 9183 or about 37% of these pairs hash into the same buckets. These do not result in errors but rather fail to decrease the number of candidates that unnecessarily require verification using the cosine similarity measure because they were not pruned.

Determining co-location requires efficient real-time computation of hashes and similarities, but it does not have as high a demand for storage as traditional media database search. Our preliminary effort indicates that LSH has potential to help scale the most demanding applications of silence signatures.

6. LIMITATIONS AND FUTURE WORK

While we believe that sensing co-location by matching patterns of relative audio silence on mobile devices is becoming practical, there are still issues for us to resolve and much more work to be done.

6.1 Limitations

Here we list known limitations of our approach.

6.1.1 Audio Drops

A high load on the mobile device CPU can cause drops in audio capture. When comparing a window of a silence signature, if the aggregate drop in the window is small compared to the average silence period, our method will still perform well. For significant drops however, a client's co-location cannot be re-established until half a window's time after the end of the period of audio dropping. Since the device is often unusable anyway during periods of such high CPU load, this might not be the most important immediate problem. We did not experience significant audio drops in our tests.

6.1.2 Participants on Mute

Employees of geographically distributed enterprises often meet via teleconferences. If the meeting involves a large enough group, some participants attending by phone will mute their phone lines so that other attendees need not suffer hearing their sneezing, snide comments, or chaotic side conversations. A participant on mute will thus only be considered co-located as long as he is quiet compared to the sounds coming over his phone. As soon as there are audible conversations in his environment, he will be considered disassociated from the rest of the group. (This extends separately to multiple different participants joining while muted.) If the conversations local to the muted participant are short and not too frequent, then the hysteresis of leaving a group often naturally covers these episodes, and so the muted participant does not flicker in and out. If the local conversations are frequent or of long duration, then the application using our silence signature matching will need another, probably explicit mechanism for the remote attendee to maintain his group membership. One possible solution to this problem is to detect a silence signature as a subset of another silence signature. We hope to work on this in the future.

6.1.3 Participants with Earphones

Another problem occurs if a participant is listening via headphones that do not allow his mobile device to capture the meeting conversations. In this situation, the silence signature his device generates will not represent the meeting to which he is listening. To circumvent this problem, we would need to provide an audio path from the device speaker to its microphone. We have not yet worked on this problem.

6.1.4 Shhh – No Talking!

There are rare circumstances where people gather together and there is no talking or significant other acoustic activity for long periods of time. These circumstances might include old-fashioned quiet libraries, peculiarly disciplined exam rooms, and monasteries for orders such as the Carthusians for whom it is important to live a life of silence during most times of the day [4]. If there are no sufficiently loud acoustic events in the environment, and if we do not create those acoustic events deliberately (see below), then our method will not work.

6.2 Future Work

Here we list some of the many problems and features we would like to tackle as part of this project.

6.2.1 Multi-modal Sensing

Like many other kinds of sensor data, we believe that silence signatures may be useful combined with other modes of sensing. For instance, Virtual Compass [9] provides positional information for nearby devices, and we have worked on gathering this information visually [21]. We would like to feed positional information to our current draggable people icons so they automatically assume the correct positions with respect to each other.

6.2.2 People at the Periphery

Our technique does not work based on set distances, but instead works by sensing which devices hear the same thing. In meetings in environments where attendees cannot easily understand who else might hear them, there may be people around who are not a part of the intended group but whose audio may pick up enough of the conversation to sense them as co-located. Applications can include facilities to allow users to pick and choose which sensed people to consider

co-located. We hope to add this as a layer to the interface to our service but have not done so yet.

6.2.3 Active Beaconing

Several of the technologies and applications mentioned in Section 2 use actively generated sound or ultrasound to perform their tasks. We too can apply this idea, for instance in silent situations such as the "no talking" environments described above, where no other devices have been detected for a long time. Devices could generate an ultrasound signal that can be sensed by other nearby devices without interfering with human perception of silence (although some current phones filter for human audible sounds).

6.2.4 Local Infrastructure

One of our future directions is embedding our approach into the infrastructure of enterprise meeting rooms. To start with, this can be as simple as a mobile phone connected to power and network and attached with Velcro to a table. This will allow the meeting rooms to determine when people are meeting. We can feed this information into our internal application for room scheduling, so it will be easier to find empty meeting rooms quickly. We can also interface meeting room displays and projectors to our service so that these devices appear as part of the group of attendees. We would then add the facilities to our content-sharing application to display content on these resources with the drag of a finger.

6.2.5 Broadcast Events

Radio and TV organizations have expressed interest in sensing when participants are attending the same broadcast event to form ad hoc social networks and provide other services (doubtless including targeted advertising) for the attendees. Our service is potentially one way to do this.

7. CONCLUSION

We have developed a technique for automatically sensing the dynamically changing membership of a meeting by using sound captured on attendees' mobile devices. From this sound, we generate a *silence signature* on each mobile device. The devices send these signatures to a cloud service that compares them. Those that match are considered part of the group, and the service sends this list back to the appropriate applications requesting the service. The technique performs robustly across a variety of real-world tests. We look forward to making this service available for purposes beyond our content sharing application and suspect that combining it with other kinds of location sensing technologies may prove useful.

8. ACKNOWLEDGEMENTS

We would like to thank the anonymous reviewers and our paper shepherd, Junehwa Song, for their many useful comments and suggestions. We also thank Kyu-Han Kim, Mostafa Uddin, JK Lee, and Yifei Jiang for answering power measurement questions, Kave Eshghi for getting us started with location-sensitive hashing, and Souvik Sen for answering questions about location sensing technologies.

9. REFERENCES

[1] Bump. http://bu.mp/company, November 2012.

[2] Foursquare. http://foursquare.com, December 2012.

[3] HP Cloud Services. https://www.hpcloud.com/, December 2012.

[4] The Carthusian Order: The Carthusian Way. http://www.chartreux.org/en/carthusian-way.php, December 2012.

[5] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM*, 51(1):117–122, 2008.

[6] Animal Systems. chirp.io. http://chirp.io, 2012.

[7] M. Azizyan, I. Constandache, and R. R. Choudhury. Surroundsense: Mobile phone localization via ambience fingerprinting. In *The 15th Annual International Conference on Mobile Computing and Networking, MobiCom 2009*. ACM, 2009.

[8] P. Bahl and V. N. Padmanabhan. Radar: an in-building rf-based user location and tracking system. In *The 19th Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM 2000*. IEEE, 2000.

[9] N. Banerjee, S. Agarwal, P. Bahl, R. Chandra, A. Wolman, and M. Corner. Virtual compass: Relative positioning to sense mobile social interactions. In *Proceedings of the 8th International Conference on Pervasive Computing*, 2010.

[10] Y.-C. Cheng, Y. Chawathe, A. LaMarca, and J. Krumm. Accuracy characterization for metropolitan-scale wi-fi localization. In *The 3rd International Conference on Mobile Systems, Applications and Services, MobiSys 2005*. USENIX, ACM, 2005.

[11] K. Chintalapudi, A. Iyer, and V. Padmanabhan. Indoor localization without the pain. In *The 16th Annual International Conference on Mobile Computing and Networking, MobiCom 2010*. ACM, 2010.

[12] M. Chong and H. Gellersen. Usability classification for spontaneous device association. *Personal and Ubiquitous Computing*, 16(1):77–89, 2012.

[13] M. K. Chong, F. Kawsar, and H. Gellersen. Spatial co-location for device association: the connected object way. In *The 2011 international workshop on networking and object memories for the internet of things, NoME-IoT 2011*, 2011.

[14] J. Chung, M. Donahoe, C. Schmandt, I.-J. Kim, P. Razavai, and M. Wiseman. Indoor location sensing using geo-magnetism. In *The 9th International Conference on Mobile Systems, Applications and Services, MobiSys 2011*. ACM, 2011.

[15] I. Constandache, R. R. Choudhury, and I. Rhee. Towards mobile phone localization without war-driving. In *The 24th Annual Joint Confernece of the IEEE Computer and Communications Societies, INFOCOM 2010*. IEEE, 2010.

[16] I. Constandache, S. Gaonkar, M. Sayler, R. R. Choudhury, and L. Cox. Energy-efficient localization for mobile phones. In *IEEE Infocom Mini Conference, 2009*. IEEE, 2009.

[17] K. Eshghi and S. Rajaram. Locality sensitive hash functions based on concomitant rank order statistics. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and*

data mining, pages 221–229. ACM, 2008.

[18] A. Gionis, P. Indyk, R. Motwani, et al. Similarity search in high dimensions via hashing. In *Proceedings of the International Conference on Very Large Data Bases*, pages 518–529, 1999.

[19] M. Goodrich, M. Sirivianos, J. Solis, C. Soriente, G. Tsudik, and E. Uzun. Using audio in secure device pairing. *International Journal of Security and Networks*, 4(1):57–68, 2009.

[20] K. Hinckley. Synchronous gestures for multiple persons and computers. In *The 16th Annual ACM Symposium on User Interface Software and Technology, UIST 2003*. ACM, 2003.

[21] W. Hong, R. Samadani, and M. Baker. Line-of-sight signaling and positioning for mobile devices. In *2nd International Workshop on Intelligent Mobile Vision (IMV2013)*, 2013.

[22] ITU-T. A silence compression scheme for G.729 optimized for terminals conforming to Recommendation V.70, Annex B, 1996.

[23] ITU-T. Coding of speech at 8 kbit/s using conjugate-structure algebraic-code-excited linear-prediction (CS-ACELP), 1996.

[24] T. Kinnunen, E. Karpov, and P. Franti. Real-time speaker identification and verification. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(1):277–288, 2006.

[25] C. Lopes and P. Aguiar. Acoustic modems for ubiquitous computing. *IEEE Pervasive Computing Magazine*, 2(3), 2003.

[26] H. Lu, A. J. Brush, B. Priyantha, A. Karlson, and J. Liu. Speakersense: energy efficient unobtrusive speaker identification on mobile phones. In *Proceedings of the 9th international conference on Pervasive computing*, 2011.

[27] H. Lu, W. Pan, N. Lane, T. Choudhury, and A. Campbell. Soundsense: Scalable sound sensing for people-centric applications on mobile phones. In *Proceedings of the 7th Annual International Conference on Mobile Systems, Applications, and Services*, 2009.

[28] R. Mayrhofer and H. Gellersen. Shake well before use: Intuitive and secure pairing of mobile devices. *Mobile Computing, IEEE Transactions on*, 8(6):792–806, 2009.

[29] P. Misra and P. Enge. *Global Positioning System. Signals, Measurements and Performance*. Ganga-Jamuna Press, 2006.

[30] T. Nakakura, Y. Sumi, and T. Nishida. Neary: Conversation field detection based on similarity of auditory situation. In *Proceedings of the 10th Workshop on Mobile Computing Systems and Applications*, 2009.

[31] A. Oppenheim and R. Schafer. *Discrete-Time Signal Processing*. Prentice Hall, 3rd edition, 2009.

[32] S. Y. Park, S. C. Jung, Y. S. Song, and H. J. Kim. Mobile robot localization in indoor environment using scale-invariant visual landmarks. In *IAPR Workshop Cognitive Information Processing*, 2008.

[33] L. Paulevé, H. Jégou, and L. Amsaleg. Locality sensitive hashing: A comparison of hash function types and querying mechanisms. *Pattern Recognition Letters*, 31(11):1348–1358, 2010.

[34] C. Peng, G. Shen, Y. Zhang, Y. Li, and K. Tan. Beepbeep: a high accuracy acoustic ranging system using cots mobile devices. In *Proceedings of the 5th international conference on Embedded networked sensor systems*, 2007.

[35] N. Priyantha, A. Chakraborty, and H. Balakrishnan. The cricket location-support system. In *The 6th Annual International Conference on Mobile Computing and Networking, Mobicom 2000*. ACM, 2000.

[36] H. Samet. *Foundations of multidimensional and metric data structures*. Morgan Kaufmann, 2006.

[37] S. Sen, B. Radunovic, R. R. Choudhury, and T. Minka. You are facing the mona lisa: Spot localization using phy layer information. In *The 10th International Conference on Mobile Systems, Applications and Services, MobiSys 2012*. ACM, 2012.

[38] G. Shakhnarovich, T. Darrell, and P. Indyk. Nearest-neighbor methods in learning and vision: Theory and practice (neural information processing), 2006.

[39] W.-T. Tan, R. Samadani, B. Lee, and M. Baker. Determining co-location using a sequential hypothesis test on patterns of silence. In *Proceedings of ICASSP 2013*. IEEE, May 2013.

[40] S. Tarzia, P. Dinda, R. Dick, and G. Memik. Indoor localization without infrastructure using the acoustic background spectrum. In *Proceedings of the 9th international conference on Mobile systems, applications, and services*, 2011.

[41] A. Wagner, J. Wright, A. Ganesh, Z. Zhou, H. Mobahi, and Y. Ma. Toward a practical face recognition system: Robust alignment and illumination by sparse representation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(2):372–386, 2012.

[42] William J. Hughes Technical Center WAAS Test Team. Civil Report Card on GPS Performance: September 2012. www.nstb.tc.faa.gov/reports/ReportCards/2012%2009.pdf.

[43] D. Wyatt, T. Choudhury, J. Bilmes, and H. Kautz. A privacy-sensitive approach to modeling multi-person conversations. In *Proceedings of the 20th international joint conference on Artifical intelligence*, 2007.

[44] M. Youssef and A. Agrawala. The horus wlan location determination system. In *The 3rd International Conference on Mobile Systems, Applications and Services, MobiSys 2005*. USENIX,ACM, 2005.

[45] B. Zhang and M. Trott. Reference-free audio matching for rendezvous. In *ICASSP*, pages 3570–3573, March 2010.

[46] J. Zhu, K. Zeng, K.-H. Kim, and P. Mhapatra. Improving crowd-sourced wi-fi localization systems using bluetooth beacons. In *IEEE International Conference on Sensor, Mesh, and Ad hoc Communication Networks, SECON 2012*. IEEE, 2012.