Low Cost Crowd Counting using Audio Tones

Pravein Govindan Kannan[‡],Seshadri Padmanabha Venkatagiri[‡], Mun Choon Chan[‡],
Akhihebbal L. Ananda[‡] and Li-Shiuan Peh[†]

[‡]School of Computing, National University of Singapore
[†]Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology
[‡]{pravein,padmanab,chanmc,ananda}@comp.nus.edu.sg, [†]peh@csail.mit.edu

Abstract

With mobile devices becoming ubiquitous, collaborative applications have become increasingly pervasive. In these applications, there is a strong need to obtain a count of the number of mobile devices present in an area, as it closely approximates the size of the crowd. Ideally, a crowd counting solution should be easy to deploy, scalable, energy efficient, be minimally intrusive to the user and reasonably accurate. Existing solutions using data communication or RFID do not meet these criteria. In this paper, we propose a crowd counting solution based on audio tones, leveraging the microphones and speaker phones that are commonly available on most phones, tackling all the above criteria. We have implemented our solution on 25 Android phones and run several experiments at a bus stop, aboard a bus, within a cafeteria and a classroom. Experimental evaluations show that we are able to achieve up to 90% accuracy and consume 81% less energy than the WiFi interface in idle mode.

Categories and Subject Descriptors

K.8.0 [**General**]: Counting; I.4.3 [**Audio processing**]: Counting; I.3.1 [**Audio**]: Speaker/Microphones

General Terms

Design, mobile audio system, experimentation

Keywords

Audio processing, tone counting, simple tones, speakers/microphones

1 Introduction

In recent years, two major trends in mobile computing have substantially changed its landscape. First, mobile devices are increasingly becoming an integral part of personal items carried by people. Second, there has been a proliferation of collaborative, crowd-sourcing based applications. Such applications include public transportation planning, real-time opinion survey, event planning and proximity marketing (refer to Section 2 for more details). In these applications, there is a strong need for a count of the number of mobile devices present in an area, as it closely approximates the size of the crowd.

Ideally, a crowd counting solution should meet the following criteria: (1) Ease of deployment: A low cost solution that leverages existing infrastructure, without requiring the installation of new sensors makes crowd counting accessible to a wider range of applications. (2) Scalability: The solution should work across large geographically regions, indoors and outdoors, and scale to large number of devices for counting huge crowds. (3) Energy efficiency: Since mobile devices are highly constrained by their battery lifetimes, good energy efficiency is needed. (4) Minimal intrusion: A solution that can effectively count in the background, with minimal involvement of the mobile device and user, will help to preserve anonymity, privacy and security. (5) Accuracy: While a very precise count is typically not required, a fairly good estimate is necessary for most applications.

Existing approaches include the use of wireless data communications (e.g. using 3G, WiFi) or RFID readers/tags. With ubiquitous smartphones, the former approach would be feasible and easily deployed. However, depending on the network technology chosen, it may not satisfy one or more of our criteria. For example, use of 3G requires infrastructure access and is not energy efficient for the purpose of counting, a very low bit-rate operation; WiFi is similarly not energy efficient; the discovery and formation of piconet/scatter net in Bluetooth is expensive and incurs substantial time delay which limits the scalability of the solution; ZigBee support on smartphones is not widespread, limiting large scale deployment. In addition, all these counting approaches require data communication which may compromise anonymity, privacy and security. These communication modes leave open multiple entry points through which mobile devices could be attacked ([9], [11]) or their privacy compromised ([8], [2]). This is because even the lowest bandwidth offered by these technologies is sufficient for intrusion and attack. Concerns with such threats ([5]) and other considerations such as power have severely limited the adoption of these approaches. Counting using RFID technology does not violate anonymity, privacy and security. However, RFID communication range is very short and only single-hop communication is supported. RFID readers are also expensive.

In this paper, we propose a crowd counting solution based on audio tones, leveraging the speakerphones that are commonly available in most phones. To the best of our knowledge, this is the first use of audio tones as a networking mechanism for multi-hop, large-scale counting.

We have implemented the tone-counting system in Android based smartphones and demonstrated that the proposed system effectively satisfies all the above criteria:

- Ease of deployment: Our solution requires just speakerphones which are widely available on most phone models, not limited to smartphones with sophisticated features.

- Scalability: The maximum number of nodes that can be counted increases exponentially with the number of frequencies available for use. Our current implementation, with 98 usable frequencies for counting, supports up to 891 devices. With the use of multi-hop communication, the devices can cover a much larger geographical area than the range of an audio transmission. Assuming an audio transmission range of 5m, 891 devices forming a 28x28 grid pattern can cover an area of 19,600 m^2 or about the size of 3.5 football fields. We have experimentally demonstrated counting in a network of up to 7 hops.

- Energy efficiency: Our power measurements show that the counting app is highly energy efficient. When operating at full capacity, our app consumes 88 mW, 82% and 91% less than using the WiFi (480 mW) and 3G (952 mW) interfaces respectively. When power saving mechanisms are enabled, WiFi consumes 57 mW with no activity, and our apps can easily be configured to operate at a mode that consumes only 10.8 mW, a 81% savings over WiFi with no activity.

- Minimal intrusion: By using audio tones which are barely audible to humans, and not requiring any user input, our solution has minimal intrusion to the mobile user experience. It also preserves anonymity, privacy, and security, since only (randomly generated) tones are sent by the phones with no additional exchange of link or MAC layer information.

- Accuracy: In our experiments using up to 25 smartphones in 3 different settings, we are able to achieve accuracy of up to 90%.

The paper is organized as follows. Section 2 discusses the motivation and Section 3 discusses the design of the system and the protocols used for tone counting. Sections 4 and 5 present the implementation and evaluation. Section 6 discusses related work and Section 7 the limitations of our work. Section 8 concludes the paper.

2 Motivation: Potential Crowd Counting Applications

We motivate our work by pointing out several potential applications where there is a need for counting:

- **Public transport planning:** Fast, low-cost estimations of the number of passengers who board a bus or subway can greatly aid public transport planning. Crowd counts localized to specific subway cabins can be leveraged for crowd control. Counts of taxi queues can guide better deployment of taxis towards high demand areas. A counting solution can also be expanded to a mechanism for binary yes/no answers such as survey questions like: "Are you a senior citizen?", "Do you get down at this stop?" etc. An effective, low-cost solution like ours can substantially expand the reach of transportation surveys, which in the past, can only be done rarely to mitigate costs [3].

  In this paper, we deployed two experiments counting the crowd at a bus stop as well as aboard a moving bus.

- **Event planning:** Events such as receptions, conferences and exhibitions will benefit from crowd counts of specific areas. For instance, a current count of the crowd at specific exhibition booths can help guide directed marketing efforts, while a count of the people at a reception can assist the event manager in ensuring sufficient service personnel is on hand to handle the guests, or arrange buffet tables to better improve service level in ergonomics or comfort.

- **Visitor Survey/Proximity marketing:** Public spaces may have information kiosks displaying advertisements. The ability to estimate the number of visitors or track the number of interested customers nearby [4] can be used to manage and organize the information displayed.

3 System Design

In this section, we first present an overview of our tone counting solution, followed by detailed description of the tone counting algorithms.

Operationally, our design requires that each mobile device, e.g. a smartphone, is able to generate one or more simple tones, and then output these tones through a speaker. A simple tone, or pure tone, refers to an audio signal with a sinusoidal waveform that can be interpreted in the frequency domain as consisting of just a single frequency.

Sampling of the signal is done at 44KHz and the frequency range detected is from 0 to 22KHz. Most speakerphones support the frequency range between 20Hz to 20KHz. Depending on age and other factors like prolonged exposure to loud noise, frequencies above 15KHz are generally not audible. In fact, MP3 supports only up to 16KHz in one of its higher compression scheme.

While transmitting, the device can also receive audio samples through its microphone. These audio samples can be processed using Fast Fourier Transform (FFT) to extract the simple tones that are transmitted by other phones and/or itself. In other words, the operations are duplex, and based on Frequency Division Multiplexing (FDM). With FDM, multiple devices can transmit multiple frequencies simultaneously without interference problems. In addition, since only simple tones are transmitted, even if the same frequency is being transmitted by multiple phones, the tone can be received correctly.

The basic mechanism is as follow. Each device starts with an initial bit pattern, and each bit corresponds to a simple

tone. In each cycle, a device broadcasts the simple tones indicated by its stored bit pattern. At the same time, it records the audio samples received and performs FFT computation periodically to recover the simple tones transmitted by other devices. These received tones correspond to the set of received bit patterns. A new bit pattern is then obtained by performing a bit-wise OR operation of the stored and received bit patterns. The transmit/receive and compute/decode cycle repeats until counting is done. Counting algorithms differ by having different initial bit patterns and deriving the device counts using different equations.

In this paper, we present two algorithms for tone counting: a simple algorithm based on uniform hashing, and a more complex algorithm based on geometric hashing. We will simply call the former the *uniform hashing approach* and the latter the *geometric hashing approach*.

3.1 Parameters

Both approaches have the following parameters in common:

1. **Frequency set, $\mathcal{F}$:** A set of frequencies that provides effective transmission. This set excludes those frequencies which are affected by ambient noise in the environment, as well as specific frequencies that cannot be transmitted due to specific speaker/microphone constraints.

2. **Control frequency set, $\mathcal{S}$ and Counting frequency set, $\mathcal{C}$:** A subset of frequencies in $\mathcal{F}$ are used for the purpose of sending control signals such as initiating the counting process, stopping the counting process etc. They belong to $\mathcal{S}$. The remaining frequencies can be used for representing device counts, and belong to $\mathcal{C}$. In other words, $\mathcal{C} \subset \mathcal{F} \setminus \mathcal{S}$. This is because guard bands (see below) are not included in $\mathcal{C}$.

3. **Guard band, $\mathcal{G}$:** When two frequencies that are adjacent to each other are transmitted, interference can result in ambiguity at the receiver. To avoid this, when determining $\mathcal{C}$, a gap is assumed between consecutive frequencies. For instance, suppose 15KHz and 15.05KHz are two consecutive frequencies in $\mathcal{F}$, then $\mathcal{G}$ is 50Hz.

4. **Tone width, $\mathcal{W}$:** Each frequency is transmitted for a certain duration, called tone width, which affects the detection accuracy and range of the frequency at the receiver(s).

5. **Tone amplitude, $\mathcal{A}$:** This is the amplitude of the transmitted frequency.

3.2 Uniform Hashing Approach

The flow of the protocol is as follows:

1. **Initialization:** Each device keeps a bit vector V_f of size $|\mathcal{C}|$. Each bit in V_f corresponds to a single frequency in $\mathcal{C}$. All bits in V_f are initialized to 0, except for a single bit that corresponds to the device's identifier. This mapping is obtained by hashing the identifier to a bit location in V_f.

2. **Count initiation:** One (or more) device initiates the counting process by transmitting the control frequency in $\mathcal{S}$ assigned for count initiation.
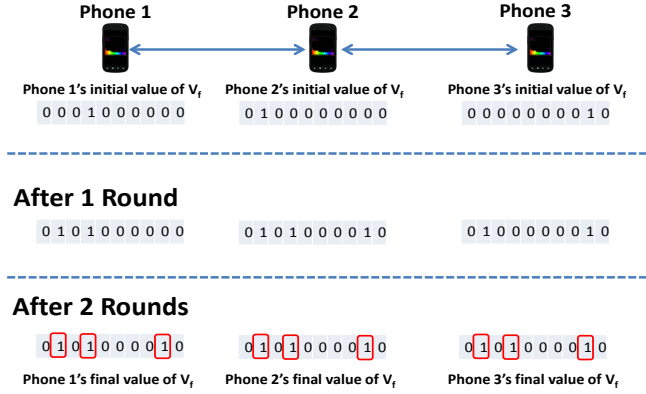

Figure 1. Illustration of uniform hashing approach with $|\mathcal{C}| = 10$.

3. **Transmission:** A frequency is transmitted if its corresponding bit location in V_f is set to 1. All such frequencies can be transmitted simultaneously. Transmission has a width of $\mathcal{W}$ and is transmitted with amplitude $\mathcal{A}$. Initially, only a single frequency corresponding to the node's identifier will be transmitted.

4. **Reception:** Reception and transmission are done in parallel (full duplex) until counting terminates. Audio samples received are stored and FFT is performed on these audio samples periodically. For each frequency detected by FFT, we set the corresponding bit in V_f to 1. If the bit is already set to 1, the frequency has already been detected. This can be thought of as a bit-wise OR operation. There is thus a link between transmission and reception. As new frequencies are detected, the corresponding bits in V_f will be set. When V_f is updated, these newly detected frequencies are transmitted. As these additional frequencies are retransmitted by neighbouring devices, information is propagated over multi-hops providing information on device count to all nodes.

5. **Count Derivation:** The number of bits in V_f that are set to 1 corresponds to the number of (unique) frequencies detected. This number is the estimated device count.

6. **Stopping Rule:** Counting stops (1) when a node hears a control frequency in $\mathcal{S}$ assigned to stop the counting process, (2) if the count stops increasing after a threshold period or (3) after a maximum counting period. To improve accuracy, we have adopted dynamic adjustment for count stabilizaton duration. Starting with an initial duration, we increase this duration if the current count exceeds a threshold value. This is because a larger stabilization duration is needed when there is a larger number of nodes to be counted.

Figure 1 shows the evolution of V_f stored on 3 phones with $|\mathcal{C}| = 10$. Initially, each phone sets a single bit in its own copy of V_f. Each phone then transmits the frequency corresponding to the bit pattern of its V_f. After 1 round of transmission, phone 2 detects the frequencies transmitted by

phones 1 and 3, while phones 1 and 3 only receive the frequency transmitted by phone 2. The bit patterns of V_fs are updated accordingly. After the second round of transmission, the bit patterns stored on all 3 phones are the same after additional frequencies are detected and V_f updated. The estimated device count is the number of bits in V_f that are set to 1, which is 3 in this example.

While simple, the uniform hashing approach clearly has its limitations. In particular, since frequencies are locally generated, multiple nodes may choose the same frequencies, leading to an under-count. The inaccuracy caused by the frequency duplication problem can be mitigated by performing a more detailed probabilistic analysis that takes into account collision probabilities [14]. However, the number of frequencies that needs to be transmitted is still on the order of the total number of devices, and its accuracy depends strongly on the cardinality of $\mathcal{C}$, thus limiting its scalability.

3.3 Geometric Hashing Approach

The basic flow of this approach is similar to the uniform hashing approach, except for the following critical differences:

1. **Initialization:**

   Similar to the uniform hashing case, each device starts with a bit vector V_f of size $|\mathcal{C}|$ with all bits initialize to 0. Next, we divide the $|\mathcal{C}|$ bit vector into m segments, each of $\frac{|\mathcal{C}|}{m}$ bits.

   Now consider the operation for a single segment i, m_i. First, apply a hash function on some unique identifier to get a value x_i that is uniformly distributed between 0 and $2^{\frac{|\mathcal{C}|}{m}} - 1$. Next, derive the corresponding h_i where h_i is computed as the bit vector where the right-most zero is changed to 1 and all other bits set to 0. For example, if $x_i = 00100111_2$, $h_i = 00001000_2$, and if $x_i = 00100001_2$, $h_i = 00000010_2$.

   Finally, let $R(h_i)$ be the bit position of the right-most zero in the bit pattern of h_i. For example, if $h_i = 00001000_2$, $R(h_i) = 3$, and if $h_i = 00000010_2$, $R(h_i) = 1$.

   Note that $R(h_i)$ is a geometric hash function. To see why this is the case, note that if the numbers are uniformly distributed over 0 and $2^{\frac{|\mathcal{C}|}{m}} - 1$, then $\frac{1}{2}$ of the possible outputs of $R(h_i)$ are 0, $\frac{1}{4}$ are 1 and $\frac{1}{2^k}$ are k-1, for k = 1,2,...,n.

   The operations for all m segments are the same, except that a different hash function is used to generate x_i and then h_i used for each segment. After these operations are applied to all segments, there is at most 1 bit set per segment and up to m bits set in V_f.

   If we consider the operation for a single segment, our estimation approach based on generating geometrically distributed identifiers is similar to [29]. However, as discussed in [10], such an estimate is not robust and has a large deviation in its output value. In order to improve the accuracy, we are in fact performing m multiple esti-

mations in parallel by having m segments with independently generated bit patterns.

2. **Count initiation:** Same as uniform hashing.

3. **Transmission:** Similar to uniform hashing except that there are up to m frequencies initially.

4. **Reception:** Same as uniform hashing.

5. **Count Derivation:** Let s_i be the bit pattern of segment i in V_f and $R(s_i)$ be the bit position of the right-most zero in the bit pattern of s_i. E(R) is the average value of $R(s_i)$ over all m segments, therefore $E(R) = \frac{\sum_i R(s_i)}{m}$. Intuitively, as the number of devices increases, $E(R)$ also increases since it is more likely that bits in the lower order positions will be set.

   Device count, N, is calculated as :

   $$N = 1.2897 * 2^{E(R)} \tag{1}$$
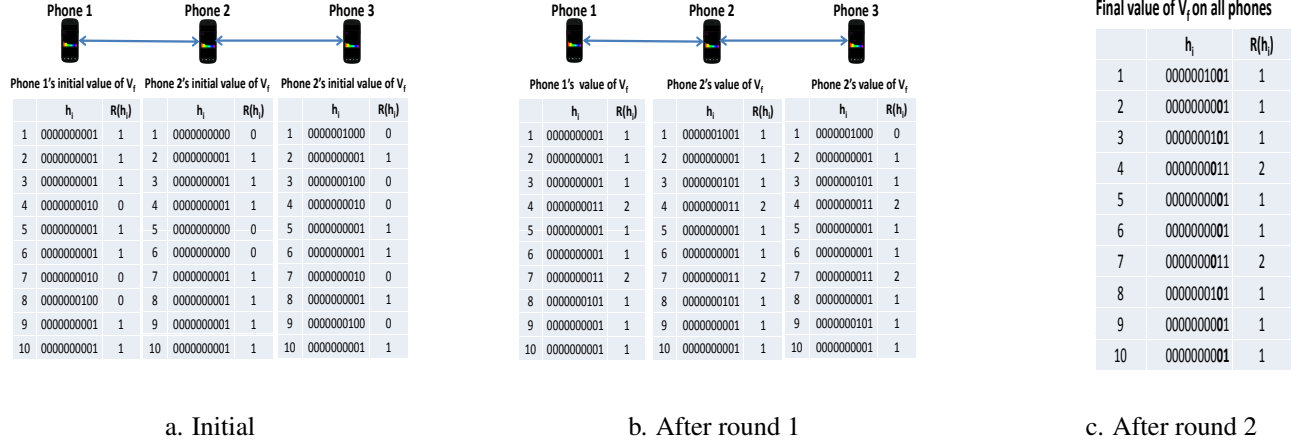
   Derivation of Equation 1 can be found in [10].

6. **Stopping Rule:** Same as uniform hashing.

Figure 2 shows the evolution of V_f on 3 phones, with $|\mathcal{C}|$ = 100 and m = 10. On each phone, 10 segments are used to perform 10 estimations in parallel. Take phone 1 for example. It hashes its identifier h into 10 different numbers h_1 to h_{10}, and set the corresponding bits in V_f. In the figure, $h_1 = 1_2$, $h_2 = 1_2$, $h_3 = 1_2$, $h_4 = 10_2$, $h_5 = 1_2$, $h_6 = 1_2$, $h_7 = 10_2$, $h_8 = 100_2$, $h_9 = 1$, and $h_{10} = 1_2$.

The phones then exchange information stored in V_f by transmitting the corresponding frequencies. After 2 rounds of information exchange, the bit patterns converge. The bit position of the right-most zero in each of the individual segments are 1, 1, 1, 2, 1, 1, 2, 1, 1, and 1. The average bit position, E(R), is $\frac{(1+1+1+2+1+1+2+1+1+1)}{10}$ = 1.2. Using Equation 1, device count is $(1.2897 * 2^{1.2})$ = 2.963.

The geometric hashing approach has a number of useful properties:

1. **Scalability in device count:** The largest number of devices that can be represented using a k-bit segment is $2^k - 1$. Hence, if $|\mathcal{C}| = 100$ and $m = 10$, k = 10. The maximum device count is 1023. There is a trade-off in terms of the maximum number of devices that can be counted and the number of estimations that can be done in parallel to improve accuracy. A larger k increases the maximum number of devices that can be counted but loses accuracy by having fewer estimates and vice versa.

2. **Reduction in frequencies transmitted:** When the number of devices is large, the geometric hashing approach requires substantially fewer frequencies to be transmitted when compared to the uniform hashing approach. For example, let $\mathcal{C} = 100$, number of devices be 200 and m is set to 10. The number of frequencies per estimate can be approximated as $E(R) = \log_2 \frac{200}{1.2897}$ = 7.277. Total number of frequencies needed for the estimation is thus 7.277*10 = 72.77, much lower than 200.

Phone 1   **Phone 2**   **Phone 3**

Phone 1's initial value of V_f   Phone 2's initial value of V_f   Phone 2's initial value of V_f

	h_i	$R(h_i)$		h_i	$R(h_i)$		h_i	$R(h_i)$
1	0000000001	1	1	0000000000	0	1	0000001000	0
2	0000000001	1	2	0000000001	1	2	0000000001	1
3	0000000001	1	3	0000000001	1	3	0000000100	0
4	0000000010	0	4	0000000001	1	4	0000000010	0
5	0000000001	1	5	0000000000	0	5	0000000001	1
6	0000000001	1	6	0000000000	0	6	0000000001	1
7	0000000010	0	7	0000000001	1	7	0000000010	0
8	0000000100	0	8	0000000001	1	8	0000000001	1
9	0000000001	1	9	0000000001	1	9	0000000100	0
10	0000000001	1	10	0000000001	1	10	0000000001	1

a. Initial

Phone 1   **Phone 2**   **Phone 3**

Phone 1's value of V_f   Phone 2's value of V_f   Phone 2's value of V_f

	h_i	$R(h_i)$		h_i	$R(h_i)$		h_i	$R(h_i)$
1	0000000001	1	1	0000001001	1	1	0000001000	0
2	0000000001	1	2	0000000001	1	2	0000000001	1
3	0000000001	1	3	0000000101	1	3	0000000101	1
4	0000000011	2	4	0000000011	2	4	0000000011	2
5	0000000001	1	5	0000000001	1	5	0000000001	1
6	0000000001	1	6	0000000001	1	6	0000000001	1
7	0000000011	2	7	0000000011	2	7	0000000011	2
8	0000000101	1	8	0000000101	1	8	0000000001	1
9	0000000001	1	9	0000000001	1	9	0000000101	1
10	0000000001	1	10	0000000001	1	10	0000000001	1

b. After round 1

Final value of V_f on all phones

	h_i	$R(h_i)$
1	0000001001	1
2	0000000001	1
3	0000000101	1
4	0000000011	2
5	0000000001	1
6	0000000001	1
7	0000000011	2
8	0000000101	1
9	0000000001	1
10	0000000001	1

c. After round 2

Figure 2. Illustration of geometric hashing approach ($\mathcal{C} = 100$, $m = 10$)

3.3.1 Comparison

Conceptually, our solution is similar to how counting can done in RFIDs [29] with some important differences. Counting in RFID system is client-server based, requires a powerful RFID reader and extends only for a single-hop. Our approach uses a peer-to-peer approach and extends over multiple hops. Another difference is that we use simple tone/frequency to represent a single bit of information and is able to transmit multiple bits/frequencies in parallel. RFID system is TDM-based and has to avoid collision. The approach described in [10] is designed for database applications and is optimized to provide an estimate by reading a large data set in a single pass.

3.4 Design space exploration through simulation

We first evaluate the two algorithms through simulation to better understand their performance and explore the design space. We simulate the exchange of tones governed by the two algorithms as described previously.

It should be noted that the simulation results presented are optimistic. In reality, (1) not all tones generated will be received, (2) there is a limit to the number of tones that a phone can transmit simultaneously due to speakerphone limitations. In the next section, we address these practical issues.

The metric chosen for evaluating the algorithms is **Error Percentage:** Error percentage, ε is given by:

$$\varepsilon = \frac{|Estimated count - Actual count| * 100.0}{Actual count} \quad (2)$$

In the simulation, we vary the following parameters. First, we vary the number of nodes/devices to be counted to understand the scalability of the algorithms from 10 to 2000. Second, we vary the total number of bits/frequencies used ($|\mathcal{C}|$) from 13 to 1664. $|\mathcal{C}|$ frequencies are used by both approaches. For the geometric hashing approach, we fix the number of bits in a segment to be 13. Hence, as $|\mathcal{C}|$ changes, we also change the number of segments or estimates used. For example, when $|\mathcal{C}| = 208$, number of segments is 208/13 = 16. Each data point was computed by running the simulation 15 times. Each run was executed with a different seed.

No. of Devices		C = 13 m = 1	C = 104 m = 8	C = 208 m = 16	C = 416 m = 32	C = 1664 m = 128
10	Uniform	27.3	2.7	0	0	0
	Geometric	18.5	20	17.2	17.5	20
100	Uniform	87	35.7	21.1	11.3	2.7
	Geometric	30.6	21.6	19.9	18.4	12.7
1000	Uniform	98.7	89.6	79.4	62.1	24.4
	Geometric	41	17.4	17.3	15.1	14.3
2000	Uniform	99.4	94.8	89.6	79.4	41.5
	Geometric	61	16.2	12.3	6.4	6.3

Table 1. Error percentage of uniform and geometric hashing with different number of bits and estimates

The results are summarized in Table 1.

As expected, the uniform hashing approach performs better when the ratio of number of nodes to number of bits used is small. For example, when the number of devices is 10, the error is less than 3% for $|\mathcal{C}| \geq 104$. However, in order to achieve high accuracy, the number of bits has to scale linearly with number of devices.

On the other hand, we can make the following observations for the geometric approach. First, this approach is much more scalable. A relatively small number of bits (e.g. 104) can be used to estimate count of up to 1000 devices with 80% accuracy. Further, increasing the number of estimates drastically does not significantly increase the accuracy. For example, when the number of estimates increases from 8 to 128, a 16 times increase in overhead, estimation accuracy increases merely from 84% to 87%. Based on this observation, we keep the number of estimates/segments in our implementation to 10.
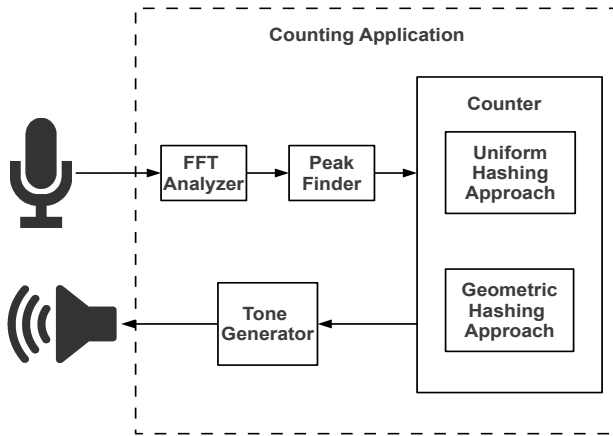
Figure 3. Components of the System

Figure 4. Peak finding methodology

4 Implementation and Baseline Measurements

4.1 Application architecture

The system consists of mobile devices which are equipped with microphone/speaker sensors. Each mobile device runs the tone counting application. The application is organized into the following components (shown in Figure 3):

1. **FFT Analyzer:** This module samples the raw audio signals from the microphone, and runs the FFT algorithm to determine the exact frequency spectrum from 0KHz to 22KHz. We use the open source Android application called Audalyzer[1] for this purpose.

2. **Peak Finder:** This module is responsible for detecting peaks in the frequency spectrum. Such peaks indicate frequencies transmitted by other nodes. The mechanism of peak detection is described in Section 4.2.

3. **Counter:** The counter module runs the algorithms discussed in Section 3. This module also chooses the frequencies to be transmitted.

4. **Tone Generator:** This module produces the frequency tones indicated by the *counter* module, and then transmits them through the speaker. For simultaneous transmission of multiple frequencies, we use *SoundPool*, a system API provided in Android 1.5.

4.2 Peak Finding Method

The detection of peak frequencies is a crucial first step in crowd counting. Specifically, we apply the following steps to the raw audio samples obtained from the microphone at a rate of 44KHz:

1. Fast Fourier Transform is used to obtain the frequency spectrum (discretized into 4096 frequency divisions) of the raw sample. The number of bins used determines the granularity of the frequencies detected.

2. An amplitude threshold is applied to all frequency samples (obtained from the FFT) above 15KHz. The am-
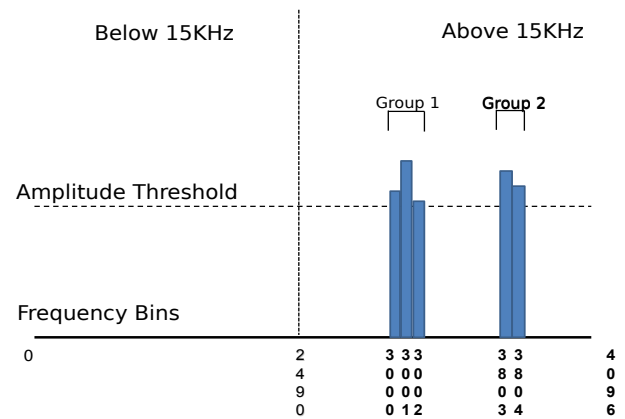
---

[1]Retrieved August 26, 2012, `http://code.google.com/p/moonblink/wiki/Audalyzer`

plitude threshold is determined based on the maximum ambient noise amplitude levels of higher frequencies. This value is set to 44dB (which is greater than the amplitude levels observed in Table 2) so that the frequency peaks could be detected with greater confidence.

3. The samples obtained from the previous step are then clustered such that successive frequency indices detected (not frequency values) are in the same group. For instance, as shown in Figure 4, if there are five frequency peaks detected, namely: 3000, 3001, 3002, 3803 and 3804, two groups are formed. One group contains {3000, 3001, 3002} and the other {3803, 3804}. The frequency corresponding to the maximum amplitude value of each group determines the peak frequency for that group. In this example, they are 3001 and 3803. Note that not all peak frequencies detected will be used as input to the counting process. Only frequencies that are deemed valid (depends on the algorithm) are used.

An important advantage of the above peak finding method is that multiple peaks can be identified at the same time. These peaks can be generated by a single mobile phone or multiple mobile phones.

4.3 Measurements

We conducted measurements in three representative environments to examine the impact of the various system parameters on the detection range, and use the result to guide our selection of the final parameters in the actual deployment. All measurements were done using the Google Nexus S phones.

We selected three environments for our measurements, 2 indoors and 1 outdoor. The first is a quiet indoor environment in an empty air-conditioned classroom. The second indoor environment is a cafeteria. We conducted our measurements between 11AM to 12PM, during the peak lunch hour. The outdoor environment chosen is a bus-stop. Cars, campus buses and public buses pass by the bus stop. We conducted our measurements between 3PM to 6PM when the vehicle traffic volume is above average. Finally, as the measurement for each set of experiments took substantial amount of time to complete, only data from the same figure are taken on the same day. As a result, even for the same configuration, the

Environment	0-14.9KHz	15-22kHz
Indoor - Lab	42dB	22dB
Indoor - Canteen	59dB	35dB
Outdoor - Bus stop	61dB	32dB
Outdoor - Bus	63dB	24dB

Table 2. Maximum noise levels in various environments

Figure 6. Plot of frequency vs. detection range

measurement result may differ slightly from figure to figure since the data may have been collected on different days.

4.3.1 Measurement of Ambient noise

First, we would like to have an indication of the ambient noise present in the various environments selected. Using the FFT Analyzer tool and Sound Meter app[2], we measured the maximum amplitude of ambient noise present in the environment since the noise level has significant impact on the values of the system parameters: $\mathcal{F}, \mathcal{S}, \mathcal{C}$ and $\mathcal{G}$ we should use. For this measurement, we have also included a fourth environment – inside a campus bus.

Plots showing the maximum amplitude of the various frequencies detected are shown in Figure 5. As expected, the noise level detected in a quiet room is the lowest. The cafeteria environment turns out to be a particularly harsh environment in terms of ambient noise because of the high frequency noise generated by the clanking of cutlery/utensils and the background noise of human conversations. For the other 2 environments, the interior of the bus is relatively quiet compared to the cafeteria or bus stop for higher frequencies. However, the engine noise creates much more low frequency noise. In all measurements, noise at lower frequencies tends to be higher than those at higher frequencies.

4.3.2 Frequency Vs Detection Range

It is well known that transmission range is a function of the frequency used. In this section, we measure the relationship between range and audio frequency with the phones used. In the measurement, we transmit one frequency at a time with tone width $\mathcal{W} = 400ms$ at maximum phone speaker amplitude. The tests are conducted in the three environments mentioned for frequencies between 12KHz to 22KHz.

The results are shown in Figure 6. The general trend is that detection range decreases with increasing frequency. Further, ambient noise plays an important role in determining the range as well. While the detection range for the indoor quiet environment reaches 8m or more for frequencies up to 20.5KHz, the range drops to 5m in the canteen and
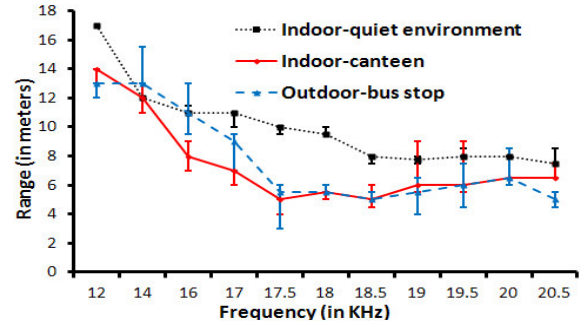
---

[2]Retrieved August 26, 2012, `https://market.android.com/details?id=kr.sira.sound&hl=en`

bus stop environments. Overall, the indoor-quiet environment has a detection range from 8 to 12 meters, the bus stop environment has a range of 5 to 10m and the cafeteria environment 5 to 8m. It can also be observed that detection range drops rapidly after 20.5KHz, most likely due to properties of speaker and microphones which do not support higher frequencies.These measurements highlighted and motivated our push towards designing a multi-hop system, in order to scale audio counting to large geographical areas, overcoming its limited range.

Based on the measurements in the last 2 sections, it is clear that we should restrict the frequency used to the range 15KHz to 20KHz. As summarized in Table 2, the maximum ambient noise in all the environments measured for frequencies between 15KHz and 22KHz ranges from 22dB (quiet room) to 35dB (canteen). On the other hand, for frequencies between 0 and 14.9KHz, the noise ranges from 42dB to 63dB. 15-22KHz comes with an additional benefit at the system level given that this is a range that is inaudible to most people, making our system less intrusive.

4.3.3 Tone Amplitude Vs Detection Range

In this experiment, we use the same tone width of $400ms$ and fix the transmitted frequency to a constant value. We varied the tone amplitude, $\mathcal{A}$ (speaker volume) and measured the detection range of the tone. As the tone amplitude decreases, the detection range decrease. We can make use of tone amplitude to restrict the neighbourhood of a particular node (see Section 7 for a further discussion). We repeat this experiment with a set of frequencies to see if the relation differs with change in frequency. The trend remains the same with a different set of frequencies. The plot for 18KHz frequency is as shown in Figure 7.

4.3.4 Tone Width Vs Detection Range

Tone width is an important parameter to consider, since tone detection depends on the sampling rate of the phone's microphone. With a smaller tone width, the likelihood of detecting the frequency decreases. We conducted the experiment with maximum amplitude and a fixed transmission frequency. We varied the tone width and measure the detection range. The measurement for 18KHz frequency is shown in Figure 8. As expected, increases in tone width translate to increases in detection range. However, using a larger tone width also stretches the time needed for counting since less

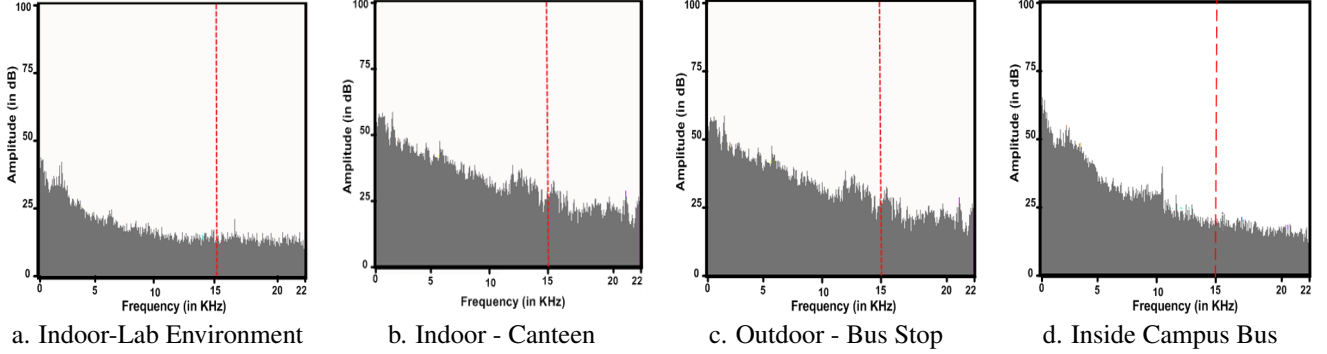| a. Indoor-Lab Environment | b. Indoor - Canteen | c. Outdoor - Bus Stop | d. Inside Campus Bus |

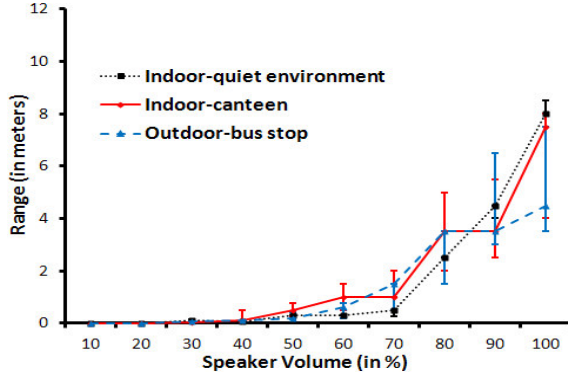**Figure 5. Ambient noise in various environments**

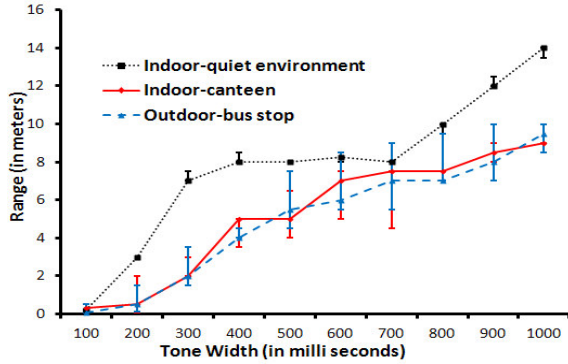**Figure 7. Plot of amplitude vs detection range (18KHz)**

Figure 8. Plot of tone width vs detection range

frequencies can be transmitted per unit time. The measurements show that detection range can be increased up to 15m in a quiet room and 9m for the other 2 environments if a tone width of 1s is used.

4.3.5 Multi-Frequency Transmission

In this experiment, we measure the performance when multiple frequencies are transmitted by 2 phones simultaneously. We measure the number of frequencies detected by 2 receivers which are placed 2 meters away from the transmitting phones and compute the percentage of tones that are correctly detected (averaged over 2 receivers).

Figure 9 shows the percentage of correctly detected tones as the number of frequency transmitted simultaneously increases, for both indoor and outdoor environments. The
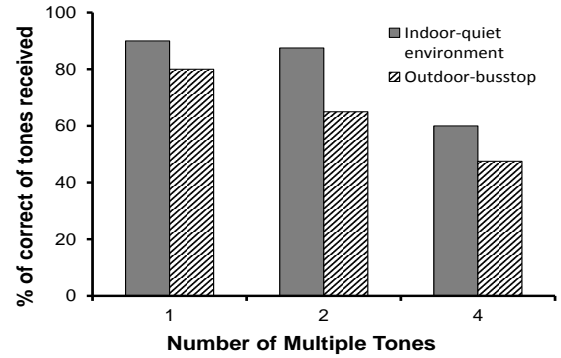

Figure 9. Percentage of correct tones received vs. number of multiple tones

percentage falls consistently with increase in number of frequency. This implies that count accuracy worsens with more simultaneous multiple frequencies transmission.

With tone reception rate as low as 60% in a noisy environment, it would seem that the accuracy of the counting algorithms would be limited. However, such a low reception rate is not as bad as it seems because of two reasons. First, there are multiple receivers. As long as one of the receivers hears the tone, the frequency is recorded. Second, our algorithm ensures that a phone transmits the same frequency multiple times.

4.4 Implementation issues

A number of practical issues affect the performance of our crowd counting solution. These issues and how they are addressed are discussed below.

4.4.1 Limitations on Speaker and Microphone Quality

The biggest challenge in the implementation is how to deal with limitations on the speaker and microphone on the smartphones. These limitations come in different forms.

First, speakers on some mobile phones (e.g. HTC Desire) introduce noise in the transmitted frequency. This noise could contribute to false peaks in the peak finder component hence affecting the accuracy of the algorithm. Certain mobile phones (e.g. Nexus One) are unable to sample frequencies above 10KHz. Such phones cannot be used for our crowd counting app. Finally, note that our transmission

scheme attempts to transmit one or more simple tones at the maximum amplitude. When too many simple tones are transmitted simultaneously, in particular, when the frequencies of these tones are close together, there may be some saturation issue in DAC, resulting in substantial noise in both transmission and reception.

For transmission, in order to reduce the noise level, we limit the number of simultaneous transmissions since when there is simultaneous transmission of multiple tones, there is a reduction in transmission range as well as increase in the noise level. However, limiting the number of simultaneous tone transmissions has a drawback; we now have to transmit the same number of tones over more rounds, which leads to the trade-off between tone width and convergence time. When tone width increases, the number of frequencies that can be transmitted per unit time decreases. Hence, we are also trading off convergence time for range and lower noise level. It is important to note that this only limits the number of tones transmitted per node per unit time. The number of tones received per unit time can be much larger if there are many neighbouring nodes transmitting.

For reception, we observe that when a tone is transmitted, the local microphone can pick up false peaks that are not received by any of the surrounding receivers. Such false peaks that are only detected by the sender are likely to be caused by transmission from the local speaker. The sender is unable to differentiate these false peaks with actual frequencies transmitted by neighbouring devices. Our measurements have shown that the frequencies of these false peaks detected by a sender are usually close to the transmitted frequency. Hence, we add the constraint that if a device is sending at a frequency f Hz, it will not accept any tone that is in the transmission frequency guard band. For example, if this guard band is 100 Hz, any frequency in the range $\{f$-100 Hz, f+100 Hz$\}$ will not be accepted by the sender.

Note that while it is true that a receiver will reject frequencies detected in the guard band even if they are actual frequencies transmitted by the neighboring nodes, it does not mean that these "rejected" frequencies will never be received. This is because the guard bands change over time with different frequencies transmitted. Hence, the frequencies that are "rejected" in one transmission cycle may be accepted in the next cycle.

4.4.2 Priority of Frequency Transmission

Due to limitation on the number of simultaneous frequencies transmitted, the order in which the frequencies are transmitted becomes important. For example, if the frequencies are transmitted from the lowest to the highest, lower frequencies will tend to be transmitted much more frequently independent of how "useful" they are.

The ideas behind our transmission scheme are as follow. First, locally generated frequencies are always transmitted first. Second, frequencies that have not been transmitted should be transmitted earlier. Third, all frequencies should be transmitted similar number of times. Finally, in order to suppress false peaks, frequencies received from neighbours are only transmitted if they have been detected at least 3 times. Once a frequency is eligible for transmission, it will be transmitted up to 5 times. Among eligible frequencies,

frequencies that have been transmitted less have higher priorities.

4.4.3 Energy Savings

While the power consumption of the speaker and microphone are low, the power consumption of FFT is substantially higher. As it is unnecessary for FFT to be executed continuously, we can implement power savings mechanisms in two ways.

First, the interval between FFT executions can be extended. Therefore, we can collect and accumulate audio samples for a longer period before running FFT to extract the frequencies.

Next, when counting is not in progress, there is no need to execute FFT. A much more power-efficient algorithm based the Goertzel algorithm [12] can be used. While FFT detects and outputs frequencies across the bandwidth of the incoming signal, the Goertzel algorithm can scan for specific frequencies. In our case, it will be looking for the presence of a small number of control frequencies. When the appropriate control frequencies are detected and counting is in progress, FFT will then be executed.

Finally, there is also no need to execute tone detection continuously but only periodically. Assuming the existence of an initiator node that broadcasts the control tone for a duration longer than the tone detection interval, the tone detection algorithm detects the control tone when counting is initiated and then starts the tone counting algorithm on demand, resulting in substantial savings. Section 5.3 presents our power measurements vs. alternative approaches.

4.5 Parameter Summary

Parameters	Value
$\mathcal{F}$	[15KHz - 20KHz]
$\mathcal{G}$	50Hz
$\mathcal{S}$	$\{15KHz, 15.05KHz\}$
$\mathcal{C}$	98 Frequencies from $\mathcal{F}$
$\mathcal{W}$	400ms
$\mathcal{A}$	100%
Tones per transmission	2
Transmission guard band	150Hz
Count stabilization time	5s to 8s
Number of Estimates	10

Table 3. Parameters used for evaluation

Table 3 lists the final parameter values chosen based on the measurement done. $\mathcal{F}$ is chosen based on the ambient noise (Figure 5) and frequency vs detection range measurements (Figure 6). The value of $\mathcal{W}$ is chosen based on measurements of tone width vs detection range (Figure 8).

The maximum number of simultaneous simple tone transmission is set to 2 because larger values introduce significant noise due to limitations of device speaker. During transmission, a node will not accept frequencies within +/- 150 Hz of the frequencies currently under transmission. Our measurements show that about 90% of the false peaks caused by speaker feedback fall into this range. We adopt multi-level stabilization based on the count value. This is because as count increases, it indicates more nodes might be potentially

available for discovery. For our scenario of 25 nodes, if the count is less than 8, then we use 5s as count stabilization time. If the count exceeds this value, we increase the stabilization to 6s. If the count exceeds 16 nodes, we accept the count if it does not change for 8s. Using a tone width of 400ms and two frequencies per transmission, each node can transmit up to 20 different frequencies in this period.

5 Evaluation

We evaluated the counting algorithms in three environments, namely: indoor-quiet (inside a air-conditioned room), outdoor-busstop and outdoor-moving-bus (inside a campus shuttle bus as described in Section 4.3).

In the indoor-quiet environment, the mobile phones were placed on desks and chairs as shown in Figure 10(a). At the bus-stop, volunteers were provided with mobile phones running our application and were asked to move around within the vicinity of the bus stop. This deployment is shown in Figure 10(b). Volunteers were asked to board a campus shuttle bus and the counting evaluation was conducted inside the bus. This is shown in Figure 10(c). As there are more phones than volunteers, some volunteers carried multiple devices.

We used 20 Google Nexus S, 5 Samsung Galaxy Nexus, 1 HTC Desire, 1 HTC Desire HD, 1 Samsung Galaxy S for our experiments. For the counting experiments, Nexus S and Samsung Galaxy Nexus phones were used. The total number of these mobile devices is 25. We used Samsung Galaxy S phone was used as initiator of the counting process. The set of phones selected for the measurement is driven by availability and not by choice.

In the evaluation, we measure the error percentage, convergence time and power consumption. Each data point is computed by taking an average of at least 3 evaluation runs. For all experiments, unless stated otherwise, the values in Table 3 are used.

5.1 Accuracy

For experiments in all 3 scenarios, we vary the number of nodes from 10 to 25. Figures 11, 12 and 13 show the results for the indoor-quiet, bus-stop and moving bus environments respectively.

As expected, the uniform hashing approach tends to performs better when the number of devices is small (e.g. 10). Over all 9 configurations (3 scenarios and 3 different counts), the errors vary from 10% to 56%. One observation from the result is that the uniform hashing approach has an inconsistent error profile with respect to the number of nodes. When the number of nodes increases, the error can increase or decrease with different environments. We believe that this is due to the fact that this approach relies directly on the accuracy of the peak finder component which is susceptible to errors introduced by the ambient noise. In the case of the quiet environment, even a small amount of ambient noise can introduce substantial error to the counting.

Overall, the geometric hashing approach performs better. Over all 9 configurations, errors vary from 3% to 35%. More importantly, the accuracy is fairly consistent over different ambient environments and tends to be better when the number of devices increases. We believe this is due to two factors. First, the use of geometric hashing makes the compu-
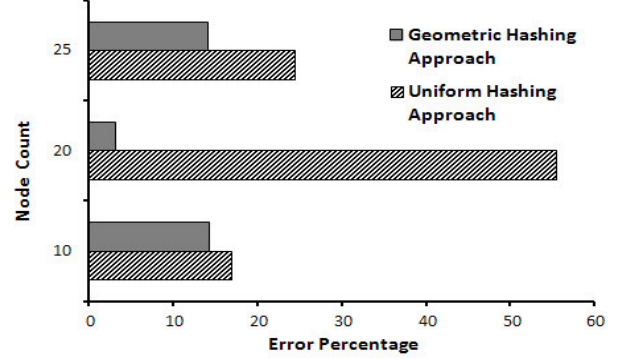

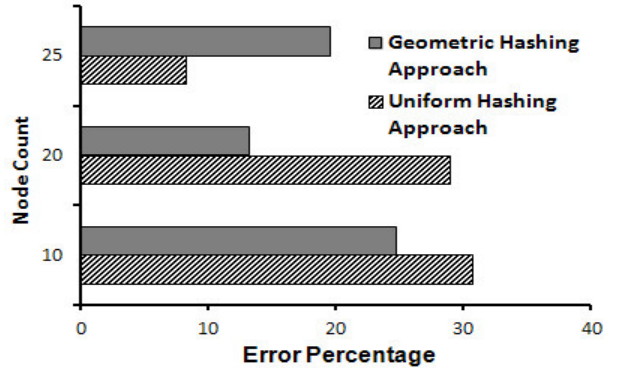**Figure 11. Evaluation results in indoor environment**

Figure 12. Evaluation results in outdoor bus-stop environment

tation less sensitive to errors introduced by the peak finding module and thus ambient noise. Second, the use of averaging which tends to reduce random errors.

As an illustration on how the individual estimates vary on the phone, Figures 14 and 15 show the cumulative distribution functions of the cases when the number of devices is 25 for the indoor-quiet and moving bus scenarios respectively. With 3 runs and 25 phones, there are a total of 75 data points. We can clearly see that while the geometric hashing approach produces estimates that are closely clustered around the actual value of 25, the uniform hashing approach tends to overestimate in the indoor-quiet scenario and underestimate in the moving bus scenario.

No. of Nodes	2	4	6	8	10	12	15
Latency (s)	0.43	6.1	7.4	8.0	7.8	8.1	7.5

Table 4. Counting latency (single hop)

5.2 Latency

Due to the much better scalability of the geometric hashing approach, we will only present latency measurement results for this approach. We record the moment that the counting process has stabilized by taking note of the time when the count has not changed for between 5s to 8s depending on the

a. Indoor-Quiet   b. Outdoor - Bus-stop   c. Outdoor - Moving Bus

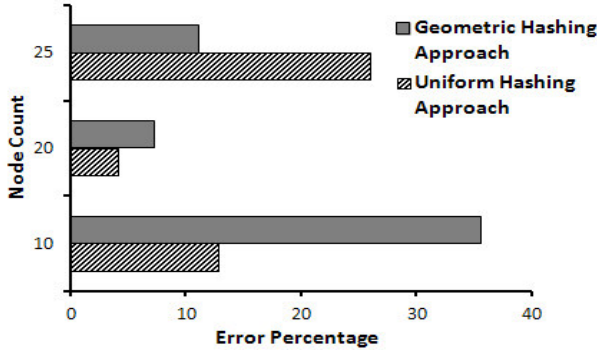**Figure 10. Evaluation in various scenarios**

Figure 15. CDF of estimated count for bus scenario (N=25)

Figure 13. Evaluation results in bus environment
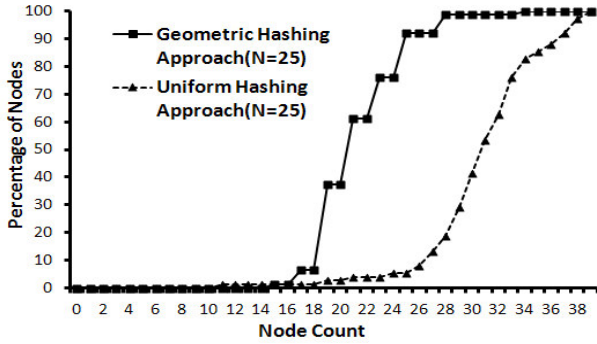
Figure 14. CDF of estimated count for indoor scenario (N=25)

Figure 16. Evaluation setup for multi-hop latency measurements

device count. We then compute the latency based on the instance at which the count is first reached. For example, if the count first reaches 10 at t=20s and stays at 10 till t=26s, latency is recorded as 20s.

We divide the latency measurements into two sets. In the first set, all nodes are within audio transmission range of one another. We increase the number of nodes from 2 to 15. The results are shown in Table 4. With only 2 nodes, there is little information that needs to be exchanged and the count con-

verges after the first audio exchange at 400ms. With more nodes, latency increases rapidly. However, the latency increases very slowly beyond 6 nodes and stabilizes at around 7s to 8s. This is because transmission and reception are performed by each node in parallel. The time for one node to transmit all the frequencies locally generated is independent of the number of nodes in total. Additional nodes do not add to the latency since all nodes can hear one another.

In the second set, we arrange nodes in a linear topology, with 2 phones per group as shown in Figure 16 . Each group forms a single cluster and can only communicate with at most two other groups in the linear topology. We increase the number of groups from 2 to 6, forming a chain topology of 1-hop to 5-hops. Table 5 shows the results. We observe that latency increases rapidly from 1-hop to 3-hops. Beyond that, the increase is more gradual. This can be explained as follow. While the increase in hop count increases the end-to-
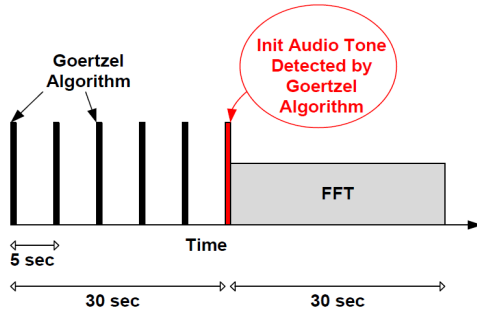
No. of Nodes	2x2	2x3	2x4	2x5	2x6
No. of Hops	1	2	3	4	5
Latency (s)	6.1	11.0	17.0	19.7	20.2

Table 5. Counting latency (multiple hops)

Figure 17. Counting process with 50% idle and 50% active

end latency, the number of additional frequencies needed for count stabilization increases very slowly (logarithmically) as the number of nodes increases. As a result, it may not be necessary to receive information from nodes that are far away for the count to become stable.

In summary, the results from the latency measurement validate two key points. First, the counting algorithm scales well with number of nodes. Second, simultaneous transmission by multiple nodes and over multiple hops work, enabling the system to cover a much bigger area beyond the range of a single node.

5.3 Power Consumption

Settings	mW
3G (ping every 10ms)	952
WiFi (ping every 10ms)	480
WiFi (ping every 100ms)	422
WiFi (ping every 1s)	65
WiFi (no activity)	57
Tone counting (FFT, continuously)	88
Tone counting (FFT, every 350ms)	73
Tone counting (FFT, every 600ms)	40
Tone detection (Goertzel, every 1s)	12
Tone detection (Goertzel, every 5s)	1.1

Table 6. Power consumption (network interface, processor and audio) for different activities

We measured the power consumed by various activities on the HTC Desire phone. For the measurement, we used the Power Tutor application[3] that measures the power consumed by the various components of the mobile device (CPU, Display, WiFi, 3G, Audio) over a period of time in Joules. The measurement results are summarized in Table 6. The power consumption shown in the table includes the network interface, processor and audio components. Power consumed by the display is not included.

We use the ping operation to emulate a counting program that reports its presence to a server through the 3G/WiFi interface. A node can periodically send its identifier to a counting server. Such an approach requires infrastructure (network access and counting server). In addition, there are pro-

---

[3]Retrieved August 26, 2012, `https://play.google.com/store/apps/details?id=edu.umich.PowerTutor&hl=en`

tocol overheads such as association to the base station, execution of DHCP to acquire IP address and possibly execution of DNS to obtain the IP address of the counting server if the IP address is not a fixed global IP address. Nevertheless, it is useful to measure and compare the energy consumptions.

When ping is performed at the very high rate of every 10ms, the 3G/WiFi interface is kept busy most of the time. Our measurements show that a counting app that runs the tone counting operations continuously consumed 88 mW as opposed to the WiFi interface and 3G interface which consume 480 mW and 952 mW respectively. This indicates that a microphone/speaker based solution can be very energy efficient, with power savings of 82% to 91% when operating at very high frequency.

Next, we measure counting operation running at a slower rate or with duty cycling including. In this measurement, we compare our approach only with WiFi. Using 802.11 infrastructure mode, we observe that the Android phone used performs aggressive power saving to reduce power consumption. When a ping program on the phone sends packets at interval of 1s, power consumption decreases to 65 mW, which is almost the same power consumption needed to keep the WiFi interface up but with no activity (57mW).

For tone counting, we observe that with FFT executing every 350ms, power consumption is 73 mW. Much higher energy savings can be achieved by operating in a duty cycle mode where a lower power tone detection operation is performed using the Goertzel algorithm. Only when the appropriate control frequency is detected will the actual tone counting operations be initiated.

As an example, let the Goertzel algorithm executes every 5s on the most recently collected audio samples accumulated. Counting is performed every minute, and runs for 30s. In the example shown in Figure 17, the Goertzel algorithm executes every 5s. In the first 30s, no control frequency is detected and the system goes into idle mode in between execution of tone detection. After 30s, the control frequency is detected and tone counting runs for the next 30s.

With a 50% duty cycle, power consumption is 0.5*1.1 + 0.5*73 = 37.1 mW, a reduction of 35% over WiFi idle power. If the counting interval is every 2 minutes and lasts for 30s, FFT executes every 600ms, and tone detection performs every 5s, energy consumption is (0.25*40 + 0.75*1.1) = 10.8mW. Savings over WiFi idle power is 81%.

The results show that the proposed tone counting approach can be very power efficient when combining with a tone detection mechanism such that counting is only performed on demand.

6 Related Work

Our solution leverages microphones and speakers for crowd counting. Here, we classify prior related works into those that use only the microphone to capture ambient sound and process it to retrieve information (*passive listening*), and those that not only listens, but also uses the speaker to transmit beacons/sound beeps (*active transmission*).

6.1 Passive listening with microphones

Environment or Traffic monitoring. The NoiseTube [22] project uses mobile phone microphones as sound sen-

sors to accumulate noise data at various locations in urban areas, creating a map of noise pollution in cities. Ear-phone [32] proposes noise pollution monitoring, using compressive sensing to fill in missing audio samples. Open issues and limitations of noise pollution monitoring using mobile devices are discussed in [33]. NeriCell [26] uses audio samples to detect horns produced by vehicles.

Social context. CenceMe [24] uses microphones to capture conversation snippets which helps infer social context such as parties. SurroundSence [6] proposes to use various mobile phone sensors such as microphone, accelerometer, camera and WiFi to identify ambient fingerprints of respective sensor data, then use them to identify the mobile user location. Neary [27] uses the intensity of perceived sound to categorize conversation fields of people.

Activity and location tracking/inference. SoundSense [18], JigSaw [19] and the framework proposed in [36] use microphone sensors for activity detection. In addition to activity detection, localization of the event source has been proposed [13], while emotions like laughter, sadness in voice can also be automatically detected [31]. Ambient noise can be used for indoor localization [35], or, along with other sensor inputs, used to modify the mobile phone profile such as the mode of operation, say, from normal mode to silent mode [34]. Darwin phones [23] and SpeakerSense [17] use mobile phone collaboration to identify the speaker using the voice data captured by the microphone. TagSense [30] uses multi-modal sensors on mobile phones to identify the environment so that pictures captured by the mobile phone could be tagged. For instance, the microphone captures voice samples which could identify whether a person in the picture is talking or laughing.

All the above proposals are passive methods which uses the microphone just for listening, instead of the two-way receive-transmit of our solution. Also, some of these proposals use customized sensor devices [16] which can limit deploy-ability.

6.2 Active transmission with microphones and speakers

Social context. In PeopleTones [15], ring tones are used as audio cues to alert an user to the proximity of his social peers. A programming interface for Symbian OS [25] can be used for ring tone based alerts or microphone sound sampling. MoVi [7] uses a ring tone (3500 Hz) produced by mobile phones to identify very small social gathering such as people sitting around a table. The proximity of the mobile phones is inferred by the similarity of the intensity of perceived sound. A central server groups all phones hearing this ring tone as belonging to the same group. The use of ring tones is very limited (use only one frequency) and not for data communication purposes.

Data transmission. Naratte, Inc.[1] proposes a single hop pairwise information exchange using microphone/speaker sensors. The communication range is 1 foot. In [20, 21], similar data transmission is proposed with a range of 2cm. However, the paper quotes a transmission experiment which achieves 8 bits/second with a range of 3.4 meters and 20 bits/second at a range of 3 meters with 0.006%

loss in indoor room environment.

Ranging. BeepBeep [28] is an acoustic-based ranging system that can operate in an ad-hoc, device-to-device context and uses COTS mobile devices. It uses the speaker and microphone as well as device-to-device communication for information exchange.

All of the above proposals are limited to interaction between a *single* pair of devices. None of them attempts a multi-hop communication between mobile devices which is necessary for scalable crowd counting. Scaling to large number of devices and multi-hop also brings about challenges in frequency allocation that is not addressed in any of the above proposals.

7 Discussion

Range Limitation and Phone Placement: An important aspect that can significantly affects tone reception range is that phones may be placed in pockets when counting is in progress. We have tried to estimate the impact of clothing on detection range and found that the detection range can decrease by as much as 50%.

We can address the range issue from the following angles. First, due to the emerging popularity of smart phone as an entertainment device, more and more users would hold the phone in their hands rather than in their pockets. Next, our system can also be adapted to run on wearable devices, like watches, smart-glasses (e.g. Google glass) that will see less obstruction. Finally, the range can be improved by using audible tones which will also increase our usable frequency range significantly. For example, there is a recent product that uses audible melodies to communicate data [4]. Use of audible tones brings about additional issues like user tolerance and robustness to ambient noise. These are interesting future directions for our work.

Scalability: We can look at the scalability of our counting approach in different ways. The main practical limitation to scalability is the size of C (counting frequency set). While it is necessary to perform multiple estimations to achieve reasonable counting accuracy, it is not practical to perform multiple sequential estimates because it will take too long to complete. There is also the need to have some form of synchronization to synchronize the start of individual counting cycle. If at least N estimations are needed, in terms of node scalability, $2^{\frac{|C|}{N}}$ is the maximum number of nodes we can count. Using the parameters chosen in our implementation, this number is $2^{\frac{98}{10}} = 891$.

In terms of geographical coverage, 891 devices can form a grid pattern of 28x28. If the grid length (tone transmission range) is 5m, the area covered is 19,600 m^2 or about the size of 3.5 football fields. If grid length is 2m, the area is 3,136 m^2, large enough to cover most subway platform.

Another parameter of interest is the time it takes for counting to converge. The network with the largest diameter possible is a linear topology where $2^{C/N}$ nodes form a line of $2^{C/N}$ - 1 hops. Consider the device at one end. If X frequencies can be transmitted using tone width of W sec,

---

[4]Retrieved August 26, 2012, `http://www.bbc.com/news/technology-18927928`

then the maximum rate of receiving new frequencies is $\frac{X}{W}$. Consider the node at one end of the topology where it can receive frequencies from only one other node. The shortest time for counting to converge in this network of $\frac{|C|}{N}$ nodes is $\frac{|C|*W}{X}$. Using the parameters chosen in our evaluation, this time is $\frac{98*0.4}{2}$ = 19.6s.

User Mobility: The tone counting algorithm works in the presence of mobility as the underlying algorithm is based on information aggregation. However, if there is a large number of nodes joining or leaving the system while counting is in progress, substantial error can be introduced. Device counting in a highly dynamic environment is not addressed in this paper and will be left for future work.

Extending to multiple counting applications: By reserving some frequencies for control purposes, one could extend the system to support multiple counting instances simultaneous or to answer simple multiple choice queries beyond those that require only simple binary response.

8   Conclusion

We have presented a counting technique based on audio tones and its evaluation in three different environments. We have also demonstrated the portability of the system by implementing the application on several models of mobile devices. Evaluation results show that it is sufficiently accurate and energy efficient. In our future work, we plan to explore how the proposed counting app can be incorporated into more complex user applications, in particular, those involving public transportation.

9   References

[1] Naratte. Retrieved August 26, 2012, http://www.naratte.com/.

[2] F-secure warns of mobile malware growth. Retrieved August 26, 2012, http://www.v3.co.uk/vnunet/news/2230481/f-secure-launches-mobile, 2008.

[3] Household Interview Survey from 1997 to 2008 A Decade of Changing Travel Behaviours. Retrieved August 26, 2012, http://www.ltaacademy.gov.sg/LA-01Journeys_May10_files/J10May-p52Choy&Toh_HIS1997-2008.pdf, 2010.

[4] 8 reasons why proximity marketing will matter for retailers in 2011. Retrieved August 26, 2012, http://www.retailcustomerexperience.com/article/178830/8-reasons-why-proximity-marketing-will-matter-for-retailers-in-2011, 2011.

[5] Mobile Threat Report. Retrieved August 26, 2012, https://www.mylookout.com/mobile-threat-report, 2011.

[6] M. Azizyan, I. Constandache, and R. Roy Choudhury. Surroundsense: mobile phone localization via ambience fingerprinting. In *MobiCom '09*, pages 261–272, 2009.

[7] X. Bao and R. Roy Choudhury. Movi: mobile phone based video highlights via collaborative sensing. In *MobiSys '10*, 2010.

[8] J. Bickford, R. O'Hare, A. Baliga, V. Ganapathy, and L. Iftode. Rootkits on smart phones: attacks, implications and opportunities. In *HotMobile '10*, pages 49–54, 2010.

[9] L. Carettoni, C. Merloni, and S. Zanero. Studying bluetooth malware propagation: The bluebag project. *IEEE Security and Privacy*, 5:17–25, 2007.

[10] P. Flajolet and G. N. Martin. Probabilistic counting algorithms for data base applications. *Journal of Computer and System Sciences*, 31(2):182–209, 1985.

[11] C. Fleizach, M. Liljenstam, P. Johansson, G. M. Voelker, and A. Mehes. Can you infect me now?: malware propagation in mobile phone networks. In *WORM '07*, pages 61–68, 2007.

[12] G. Goertzel. An algorithm for the evaluation of finite trigonometric series. 65(1):34–35, Jan 1958.

[13] Y. Guo and M. Hazas. Acoustic source localization of everyday sounds using wireless sensor networks. In *Ubicomp '10*, 2010.

[14] M. Kodialam and T. Nandagopal. Fast and reliable estimation schemes in rfid systems. In *Mobicom'06*, 2006.

[15] K. A. Li, T. Y. Sohn, S. Huang, and W. G. Griswold. Peopletones: a system for the detection and notification of buddy proximity on mobile phones. In *MobiSys '08*, 2008.

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

[17] H. Lu, A. J. B. Brush, B. Priyantha, A. K. Karlson, and J. Liu. Speakersense: energy efficient unobtrusive speaker identification on mobile phones. In *Pervasive'11*, 2011.

[18] H. Lu, W. Pan, N. D. Lane, T. Choudhury, and A. T. Campbell. Soundsense: scalable sound sensing for people-centric applications on mobile phones. In *MobiSys '09*, 2009.

[19] H. Lu, J. Yang, Z. Liu, N. D. Lane, T. Choudhury, and A. T. Campbell. The jigsaw continuous sensing engine for mobile phone applications. In *SenSys '10*, 2010.

[20] A. Madhavapeddy, D. Scott, and R. Sharp. Context-aware computing with sound. In *UbiComp 2003: Ubiquitous Computing*. 2003.

[21] A. Madhavapeddy, R. Sharp, D. Scott, and A. Tse. Audio networking: The forgotten wireless technology. *IEEE Pervasive*, 4(3):55–60, 2005.

[22] N. Maisonneuve, M. Stevens, and B. Ochab. Participatory noise pollution monitoring using mobile phones. *Info. Pol.*, 15:51–71, April 2010.

[23] E. Miluzzo, C. T. Cornelius, A. Ramaswamy, T. Choudhury, Z. Liu, and A. T. Campbell. Darwin phones: the evolution of sensing and inference on mobile phones. In *MobiSys '10*, 2010.

[24] E. Miluzzo, N. D. Lane, K. Fodor, R. Peterson, H. Lu, M. Musolesi, S. B. Eisenman, X. Zheng, and A. T. Campbell. Sensing meets mobile social networks: the design, implementation and evaluation of the cenceme application. In *SenSys '08*, 2008.

[25] A. Misra, G. Essi, and M. Rohs. Microphone as sensor in mobile phone performance. *NIME 2008*, 2008.

[26] P. Mohan, V. N. Padmanabhan, and R. Ramjee. Nericell: rich monitoring of road and traffic conditions using mobile smartphones. In *SenSys '08*, 2008.

[27] T. Nakakura, Y. Sumi, and T. Nishida. Neary: conversation field detection based on similarity of auditory situation. In *HotMobile '09*, 2009.

[28] C. Peng, G. Shen, Y. Zhang, Y. Li, and K. Tan. Beepbeep: a high accuracy acoustic ranging system using cots mobile devices. In *SenSys'07*, pages 1–14, 2007.

[29] C. Qian, H. Ngan, and Y. Liu. Cardinality estimation for large-scale rfid systems. In *PerCom'08*, pages 30–39, 2008.

[30] C. Qin, X. Bao, R. Roy Choudhury, and S. Nelakuditi. Tagsense: a smartphone-based approach to automatic image tagging. In *MobiSys '11*, 2011.

[31] K. K. Rachuri, M. Musolesi, C. Mascolo, P. J. Rentfrow, C. Longworth, and A. Aucinas. Emotionsense: a mobile phones based adaptive platform for experimental social psychology research. In *Ubicomp '10*, 2010.

[32] R. K. Rana, C. T. Chou, S. S. Kanhere, N. Bulusu, and W. Hu. Earphone: an end-to-end participatory urban noise mapping system. In *IPSN '10*, 2010.

[33] S. Santini, B. Ostermaier, and R. Adelmann. On the use of sensor nodes and mobile phones for the assessment of noise pollution levels in urban environments. In *INSS'09*, 2009.

[34] D. Siewiorek, A. Smailagic, J. Furukawa, A. Krause, N. Moraveji, K. Reiger, J. Shaffer, and F. L. Wong. Sensay: A context-aware mobile phone. In *ISWC '03*, 2003.

[35] S. P. Tarzia, P. A. Dinda, R. P. Dick, and G. Memik. Indoor localization without infrastructure using the acoustic background spectrum. In *MobiSys '11*, 2011.

[36] Y. Wang, J. Lin, M. Annavaram, Q. A. Jacobson, J. Hong, B. Krishnamachari, and N. Sadeh. A framework of energy efficient mobile sensing for automatic user state recognition. In *MobiSys '09*, 2009.