Bubble-Sensing: A New Paradigm for Binding a Sensing Task to the Physical World using Mobile Phones

Hong Lu* Nicholas D. Lane* Shane B. Eisenman† Andrew T. Campbell*

*Dartmouth College
Hanover, New Hampshire, USA
{hong,niclane,campbell}@dartmouth.edu

†Columbia University
New York, New York, USA
shane@ee.columbia.edu

***Abstract*—We propose Bubble-Sensing, a new sensor network abstraction that allows mobile phones users to create a binding between tasks (e.g., take a photo, or sample audio every hour indefinitely) and the physical world at locations of interest, that remains active for a duration set by the user. We envision mobile phones being able to affix task bubbles at places of interest and then receive sensed data as it becomes available in a delay-tolerant fashion, in essence, creating a living documentary of places of interest in the physical world. The system relies on other mobile phones that opportunistically pass through bubble-sensing locations to acquire tasks and do the sensing on behalf of the initiator, and deliver the data to the bubble-sensing server for retrieval by the user that initiated the task. We describe an implementation of the bubble-sensing system using sensor-enabled mobile phones, specifically, Nokia's N80 and N95 (with GPS, accelerometers, microphone, camera). Task bubbles are maintained at locations through the interaction of "bubble carriers", which carry the sensing task into the area of interest, and "bubble anchors", which maintain the task bubble in the area when the bubble carrier is no longer present. In our implementation, bubble carriers and bubble anchors implement a number of simple mobile-phone based protocols that refresh the task bubble state as new mobile phones move through the area. Phones communicate using the local ad hoc 802.11g radio to transfer task state and maintain the task in the region of interest. This task bubble state is ephemeral and times out when no bubble carriers or bubble anchors are in the area. Our design is resilient to periods when no mobiles pass through the bubble-area and is capable of "reloading" the task into the bubble region. In this paper, we describe the bubble-sensing system and a simple proof of concept experiment.**

I. Introduction

The mobile phone has become a ubiquitous tool for communications, computing, and increasingly, sensing. Many mobile phone and PDA models (e.g., Nokia's N95 and 5500 Sport, Apple's iPhone and iPod Touch, and Sony Ericsson's W580 and W910) commercially released over the past couple years have integrated sensors (e.g., accelerometer, camera, microphone) that can be accessed programmatically, or support access to external sensor modules connected via Bluetooth. The sensed data gathered from these devices form the basis of a number of new architectures and applications [3] [2] [1] [7] [6]. We present the Bubble-Sensing system, that acts to support the persistent sensing of a particular location, as required by user requests. Conceptually, a user with a phone that has opted into the Bubble-Sensing system visits a location of interest, presses a button on his phone to affix the sensing request to the location, and then walks away. The sensing request persists at the location until the timeout set by the initiator is reached. This mechanism can be viewed as an application in its own right (e.g., a user slogging [4] his life), and as a persistent sensing building block for other applications.

While the notion of virtually affixing sensor tasks to locations is appealing, it requires some work to implement this service on top of a cloud of human-carried phone-based sensors. First, since the mobility of the phones is uncontrolled - there is no guarantee that sensors will be well-placed to sample the desired location specified by the sensing task. Further, there is the issue of communicating the sensing task to potential sensors when they are well-positioned. This is made more difficult when, either due to hardware or user policy limitations, an always-on cellular link and localization capabilities are not available on all phones. For example, wireless data access via EDGE, 3G, or open WiFi infrastructure is increasingly available, as is the location service via on-board GPS, WiFi, or cellular tower triangulation. However, for example, only a subset of mobile phones on the market have GPS and WiFi, and even when devices have all the required capabilities, users may disable the GPS and or limit data upload via WiFi and cellular data channels to manage privacy, energy consumption, and monetary cost.

Though the mobility in a people-centric sensor network is not controllable, it is also not random. In an urban sensing scenario, the visited areas of interest for one person are likely to be visited by many others (e.g., street corners, bus/subway stations, schools, restaurants, night clubs, etc.). We imagine a heterogeneous system where users are willing to share resources and data and to fulfill sensing tasks. Therefore, the bubble-sensing system opportunistically leverages other mobile phones as they pass by on behalf of a sensing task initiator. We adopt a two tier hardware architecture comprising the bubble server on the back end; and sensor-enabled mobile phones that can initiate sensing bubbles, maintain sensing bubbles in the designated location, replace bubbles that disappear due to phone mobility, enact the sampling as indicated by the sensing bubble, and report the sensed data back to the bubble server. Mobile phones participating in the bubble-sensing system take on one or more roles depending on their mobility characteristic, hardware capabilities, and user profiles. The *bubble creator* is the device whose user initiates

the sensing request that leads to the creation of the sensing bubble. The *bubble anchor* keeps the bubble in the region of interest by broadcasting the sensing request. The *sensing node* perceives the bubble by listening to the broadcasts, takes samples within the area of interest according sensing request, and then uploads the results to the bubble server. The bubble carrier can help to restore a bubble if all bubble anchors are lost. The bubble server binds the results to the bubble, which can be queried by the bubble creator at any time.

We have implemented the bubble-sensing system using Nokia N95 mobile phones. In Section II, we describe the specific responsibilities of the virtual roles mentioned above and provide details on the communication protocols required to implement these roles. Sections III and IV describes our current implementation and a preliminary evaluation of bubble-sensing using a N95 testbed, reporting on temporal sensing coverage, and on a measure of sensed data quality. We discuss related work from the pervasive and mobile ad hoc networking communities, including comparisons to alternative implementation choices, in Section V. In Section VI, we discuss possibilities for extending the current work and offer concluding remarks.

II. Bubble-Sensing

Sensing tasks are created and maintained in the bubble-sensing system through the interaction of a number of virtual roles, where a given physical node can take on one or more virtual role based on its location, device capabilities (e.g., communication mode, sensor), user configuration (when and to what extent resources should be shared for the common good), device state (e.g., an ongoing phone call may preclude taking an audio sample for another application), and device environment (e.g., a picture taken inside the pocket may not be meaningful to the data consumer). In the bubble-sensing system, a task is a tuple $(action,\ region,\ duration)$ The $action$ can be any kind of sensing operation such as "take a photo", or "record a sound/video clip". The $region$ is defined as the tuple $(location,\ radius)$, where $location$ is a point in a coordinate system like GPS indicating the center of the region, and the $radius$ defines the area of the region. We call this region of interest the "sensing bubble". In the following, we describe each of the virtual roles (i.e., bubble creator, bubble anchor, sensing node, and bubble carrier) in the context of the major system operations: bubble creation, bubble maintenance, bubble restoration. Figure 1 gives a pictorial representation of the Bubble-Sensing architecture and the main bubble management steps.

A. Bubble Creation

The bubble creator is the device whose user initiates the sensing request that leads to the creation of the sensing bubble. Generally speaking, there are two ways a bubble can be created. In the first scenario, the creator is a mobile phone. The phone's carrier moves to the location of interest and creates the sensing task. In the second scenario, the creator is any entity that registers a task with the bubble server, but does interact with other nodes at the location of interest in support of the sensing. As the process flow for the second case is a subset of the first (c.f. bubble restoration in Section II-D), in the following we omit any further explicit discussion of the second scenario.

Proceeding with a discussion of the first scenario, we assume the bubble creator is a mobile device at the location of interest with a short range radio for local peer interactions. The creator (e.g., node *A* in Figure 1) broadcasts the sensing task using its short range radio. If the user has enabled cellular data access to the backend bubble server, the creator also registers the task with the bubble server. If the creator has localization capability, it populates the $region$ field of the task definition, and the sensing bubble is created with its center at this location. Otherwise, the $region$ field of the task is left blank in the broadcast, and the sensing bubble is created with its center at the current location of the creator where the area of the bubble is determined by its radio transmission range. Note, that if the creator is not able to obtain a location estimate and register its task with the bubble server, it will not be possible to restore the bubble later (c.f. Section II-D) in case the bubble disappears due to temporary lack of suitable mobile nodes in the area of interest. Nodes that receive the task broadcast and meet the hardware and context requirements for the sensing task can then sense in support of the task, and will later upload the sensed data to the bubble server in either a delay-tolerant (e.g., opportunistic rendezvous with an open WiFi access point), or real-time (e.g., the cellular data channel) manner.

B. Bubble Maintenance

Given the uncontrolled mobility of the creator, it may happen that the creator leaves the bubble location while the bubble task is still active (as specified in the $duration$ field of the task). If this happens, it is no longer appropriate for the creator to broadcast the task since recipients of this broadcast will not be in the target bubble location. A way to anchor the bubble to the location of interest is needed; the bubble anchor role fills this requirement (e.g., node *B* in Figure 1). The node that takes on this role should be relatively stationary at the target location of the task. We propose two variants for bubble anchor selection, one that requires localization capability on all nodes (e.g., GPS), and one that uses inference from an accelerometer for mobility detection.

1) Location-based: In the location-based approach, all nodes that find themselves in the sensing bubble with knowledge of the bubble task (i.e., they can hear the bubble task broadcasts) are potential anchor candidates. If the candidate does not hear another anchor (as indicated by a special field populated in the bubble task broadcast) for a particular threshold time, indicating the bubble is not currently covered by an anchor, it prepares to become the anchor for that bubble. Each candidate anchor backs off a time proportional to its mobility as measured by speed inferred from changes in the location fixes. After this backoff time, a candidate that does not overhear any other anchor broadcasting the task then

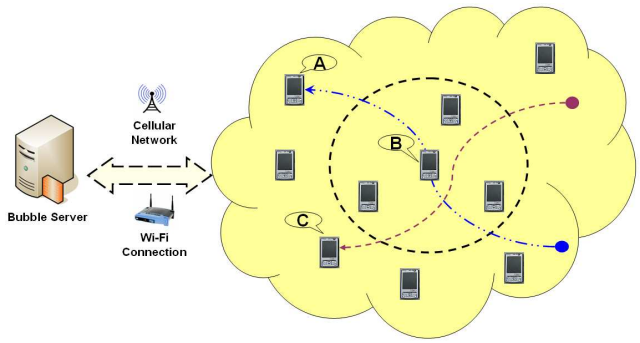

Fig. 1. Bubble-Sensing architecture and bubble management. Phone *A* is the task creator. *A* is moved by its human carrier to the area of interest, and attempts to attach the sensing bubble to the area by broadcasting the sensing task via its local radio, and also registers the task with the bubble server via its cellular radio. Stationary phone *B* receives the task broadcasts from *A* and assumes the role of bubble anchor. As the mobility of *A* takes it out of the bubble area (indicated by the dashed circle), *B* takes over the management of the sensing task by continuing to broadcast the sensing task to passersby. If the anchor *B* later moves away, the bubble temporarily disappears. A phone *C* that later moves through the area of interest is signaled by the bubble server, becomes a bubble carrier, and tries to re-affix the sensing bubble by broadcasting the task via its local radio. Sensed data gathered by phones that accept the sensing task broadcasted by the bubble creator, bubble anchor, or bubble carrier can be uploaded in real time via the cellular network, or in a delay tolerant fashion via a local radio gateway (e.g., WiFi).

assumes the role of bubble anchor. The anchor will continue to broadcast the task beacon (with the special field to indicate an anchor is sending it) until it moves out of the location of interest for that bubble.

2) Mobility-based: In the mobility-based approach, like the location-based approach, nodes that can hear the bubble task broadcasts are potential anchor candidates. If the candidate does not hear another anchor broadcasting the bubble task, it backs off a time proportional to its mobility, as inferred from data collected by its accelerometer. After this backoff time, a candidate that does not overhear any other anchor broadcasting the task then assumes the role of bubble anchor. The anchor will continue to broadcast the task beacon (with the special field to indicate an anchor is sending it) until it moves out of the location of interest for that bubble. In this case, the mobility is again determined through classification of data from the on-board accelerometer.

C. Challenges to Bubble Maintenance

The broadcast-based approach to bubble maintenance introduces two main sources of error to the data collected in support of the sensing task. First, since we do not require sensing nodes to have knowledge of their absolute location, recipients of the task broadcast that are outside of the bubble area defined in the broadcast may still collect and upload data to the bubble server. This potentially makes the effective bubble size larger than the specified bubble size. The extent of this distortion depends both on the radio range of the task broadcast, and the location of the broadcaster (i.e., bubble creator, bubble anchor, and bubble carrier - c.f. Section II-D) with respect to the specified bubble center location. If location-based bubble maintenance is used, or if the sensing node has localization capabilities, the location information may be used to compensate the transmission power of the task broadcast or suppress sensing when nodes are outside of the defined bubble area to reduce this bubble distortion.

The second source of error is bubble drift, which can happen for two reasons. First, drift can happen over time if the anchor moves but continues to broadcast the bubble task due to inaccuracy in its mobility/location-detection methods. While improvements in localization technology and mobility classification can help here, we also explicitly limit the consecutive amount of time a node can act as the anchor for a given bubble. Assuming a probabilistic mobility/location error model, it would be possible to calculate the appropriate timeout to probabilistically limit the bubble drift below a desired level. The second cause of bubble drift is limited to the mobility-based bubble maintenance method where ubiquitous localization not assumed. In this case, as the current anchor gives up its role (e.g., out of battery, or anchor role timeout, move out of the bubble region), one of other semi-stationary or slow moving nodes available in the bubble will take over the anchor role as mentioned in Section II-B. This can be viewed as a passive role handoff. However, with each handoff the center of the bubble drifts to the location of the new anchor and over time this can markedly distort the sensing coverage of the bubble. To counteract this source of drift, we implement a limit on the number of anchor handoffs. After the handoff limit is reached, the anchor must be reinitialized by the bubble restoration process described in the following. We note that if mobile devices have continuous localization capability (e.g., using GPS, GPS assisted with GSM [14], WiFi [15]), then bubble distortion and drift is limited by the localization inaccuracy.

D. Bubble Restoration

Due to node mobility, it may happen that no nodes are available to anchor the bubble to the desired location and the bubble may temporarily disappear. To address this scenario, the bubble-sensing system provides a mechanism for bubble restoration through the actions of bubble carrier nodes (e.g., node *C* in Figure 1). Mobile phones filling the bubble carrier role require localization capability and a connection to the backend bubble server. Bubble carriers periodically contact the bubble server, update their location, and request any active sensing bubbles in the current region. If a bubble carrier visits the location of one of these bubbles and does not hear any task broadcasts, it attempts to restore the bubble by broadcasting the task without the special anchor field set (in the same way the bubble creator did initially). Through this method, either the bubble will be restored with a new anchor node taking over the bubble maintenance, or this attempt at restoration fails. Bubble restoration attempts continue via the bubble carriers until the bubble expires (as indicated by the $duration$ field in the bubble task definition).

III. Implementation

We build a proof-of-concept mobile cell phone test bed to demonstrate the bubble sensing system. The test bed consists of Nokia N80 and N95 smart phones, both of which run Symbian OS S60 v3. Due to the security platform in Symbian, some hardware access APIs are restricted at the OS level and are not open to developers, or require a high privilege certificate. In light of the platform limitations on these two mobile phones, in this section, we discuss the options available and our implementation choices.

A. Programming Language

We use PyS60 (http://sourceforge.net/projects/pys60/) to prototype our system. PyS60 is Nokia's port of Python to the Symbian platform. It not only supports the standard features of Python, but also has access to the phone's functions and the on-board sensors (e.g., camera, microphone, accelerometer and GPS), software (e.g., contacts, calendar), and communications (e.g., TCP/IP, Bluetooth, and simple telephony). In addition to that, the developer can easily add access to the native Symbian APIs using the C/C++ extension module. In this regard, PyS60 is more flexible than Java J2ME in providing robust access to native sensor APIs and phone state, as we discovered in our initial development.

B. Communication

The Nokia N80 and N95 mobile phones are both equipped with GPRS/EDGE, 3G, Bluetooth and WiFi interfaces. For data uplink, they can leverage GPRS, SMS, and MMS for the universal connectivity, and WiFi/Bluetooth access points can also provide Internet access when available. For local communication, Bluetooth and WiFi are two possible choices. In our test bed, WiFi is our choice for both local communication and communication to the Internet. Considering the cost of the data service for GPRS and existing open WiFi infrastructure in the academic and urban environments, WiFi is a viable option for Internet access. To implement bubble-sensing, broadcast is fundamental and indispensable. While our initial choice for local communication was Bluetooth since it currently enjoys a higher rate of integration into mobile phones, we found peer to peer broadcast with Bluetooth technology to be particularly difficult. Fortunately, we can configure the phones to use the Ad-Hoc IEEE 802.11 mode and the UDP broadcasting over WiFi is relatively easy to use. In out current version, the phone uses Ad-Hoc mode when interacting locally with peers, and infrastructure mode to connect to the bubble server. Phones can switch between these two modes on the fly when necessary. The lag of the mode switch is as low as a few seconds. We also set the transmit power of the WiFi interface to the lowest, 4mW, in order to save energy.

C. Sensors and Classifier

Camera and microphone sensors are universal on mobile phones nowadays. In our experiment, to save storage and lower the transmission load, we use lower resolution pictures (640 $\times$ 480 pixels). For sound, we record two second sound clips in `.au` format; each sound clip is about 28 kB. All data collected are time stamped. For the accelerometer and GPS sensors, the N95 comes with an on-board GPS and a built-in 3D accelerometer. We extend the N80 using the external BluCel device (see Figure 2), basically a Bluetooth-connected 3D accelerometer. Both types of accelerometers are calibrated, and the data output are normalized to earth gravity. The sampling rate is set to 40 Hz. Phones perform some relatively simple processing on the data samples (e.g., mean, variance, and threshold) and feed the features extracted from data samples to a simple decision tree classifier, which classifies the movement of people carrying the phone. The classifier does not require the user to mount the mobile phone in a particular way; users can simply put the phone in a pocket or clip it on the belt. The tradeoff for this flexibility is that the classifier can only differentiate basic movements of people, i.e., stationary, walking, running. Complicated movements like stair-climbing and cycling will likely by classified as either walking or running. However, our system only requires the discrimination between stationary and moving. In this sense, this light weight classifier provides sufficient accuracy (see the confusion matrix in Table I).

Fig. 2. The BluCel device provides a 3D accelerometer that can be connected to the mobile phone (e.g., Nokia N95 or N80) via Bluetooth.

	Stationary	Moving
Stationary	0.9844	0.0155
Moving	0.0921	0.9079

TABLE I
THE CONFUSION MATRIX FOR OUR STATIONARY/MOVING CLASSIFIER.

D. Localization

There are many existing solutions that provide a localization service for mobile phones, including built-in/external-connected GPS, cell-tower triangulation (GSM fingerprint), Bluetooth indoor localization, and WiFi localization systems such as Skyhook and Navizon. For Symbian, to get all the cell towers information requires a high privilege certificate not available to most developers. Usually, the developer can only get the information about the cell tower to which the phone is currently connected. This does provide a rough sense of

	Static	Ideal	Limited	Location-based	Mobility-based	
					All	Out
Trial 1	158	1026	181	967	1004	78
Trial 2	143	679	165	579	607	42
Trial 3	98	324	74	294	304	40

TABLE II
SAMPLE COUNTS FOR THE FIVE SCHEMES DESCRIBED IN SECTION IV-A: STATIC, IDEAL, LIMITED, LOCATION-BASED, MOBILITY-BASED.

where the device is, but is not sufficient for the triangulation algorithm. Therefore, in the outdoor case we simply use GPS. For indoor, the WiFi fingerprint is a natural choice for academic and urban environments, given the relatively widespread coverage of WiFi infrastructure.

E. System Integration

Use of the mobile phone as a sensor in the bubble-sensing system should not interfere with the normal usage of the mobile phone. Our bubble-sensing software implementation is light weight, so users can easily switch it to background, and use their phone as usual. The software only accesses sensors on demand and release the resources immediately after use. An incoming or user-initiated voice call has high priority, and our software does not try to access the microphone when it detects a call connection. By adapting in this way, our implementation does not disrupt an ongoing call and also the bubble-sensing application will not get killed by an incoming call. We test the CPU and memory usage of our software in a Nokia N95, using a bench mark application, CPUMonitor [13]. The peak CPU usage is around 25%, which happens when sound clips are taken. Otherwise, the CPU usage is about 3%. The memory usage is below 5% of the free memory, including the overhead of the python virtual machine and all the external modules.

IV. TESTBED EVALUATION

In order to evaluate our implementation of the bubble-sensing system, we perform a series of indoor experiments. The aim of this evaluation is to validate the performance of a mobile cell phone network and how it can benefit from the use of bubble sensing mechanisms, mainly in terms of the number of data samples collected and the time coverage of those samples.

A. Experiment Setup

Ten mobile phones are carried by people who move around three floors of the Dartmouth computer science building. The carriers stay mobile for the duration of the experiment, except for momentary pauses at the water cooler, printer, or desk (to check for important emails). No particular effort is made to orchestrate the mobility to maintain density in the sensing bubble or elsewhere. The participants are told to carry the cell phones as they normally would. Most of the time the mobile phones are put in the front or back pockets and sometime held in the hand (e.g., when making a call, checking the time, sensing a SMS message, etc). Static beacons are used to provide a WiFi localization service. In our experiments, the center of the task bubble is defined to be the Sensor Lab, which is a room on the middle of the three floors. The task is assumed to already be registered by the bubble creator. During the experiment, we play music in the bubble and the task is simply capturing sound clips in this room once every ten seconds. To emulate a heterogeneous network, we intentionally limit device capabilities (i.e., long range connectivity and localization) in some cases. We evaluate the following five different cases:

- **Static sensor network.** for comparison, we deploy one static sensor node (a N95) in the center of the bubble, programmed to periodically do the sensing. The static node is about three meters away from the source of the music, a pair of speakers, and the microphone is pointed in the direction of the music source.
- **Ideal mobile sensor network.** Mobile nodes in the network always have cellular data uplink and localization and can therefore always retrieve the bubble task from the bubble server, and can tell when to do the sensing. No bubble sensing techniques are used; in fact none are required since all nodes know about all bubble tasks in the system. The results of this case represent an upper bound on what can be expected in the system when using mobile sensors.
- **Limited-capability mobile sensor network.** Assuming universal always-on connectivity is unrealistic, for both technological and social reasons. Many are unwilling to pay the extra monthly charge to add data service to their cellular service package. In urban environments, such as New York City, we experience frequent dropped connections even outdoors at street level due to interference and fading. Cellular reception indoors is even more inconsistent due to signal attenuation. In rural environments like Hanover, NH, we experience frequent dropped connections due to borderline coverage. Here we make the more realistic assumption that mobile nodes have only a 0.25 probability of an available data uplink (ability to fetch the task from the bubble server and do the sensing) at the moment when they enter the bubble. In this scenario, all nodes are still assumed to have the capability for localization. Again, no bubble-sensing techniques are used.
- **Bubble-sensing with location-based bubble maintenance.** This scenario builds on the limited-capability mobile sensor network case by adding bubble carrier and bubble anchor functionality. Therefore, any nodes they hear task broadcasts and are in the bubble will do the sensing. Bubbles are maintained using the location-based scheme (universal localization capability assumed), and are restored using bubble carriers. Mobile nodes entering the bubble location become task carriers with a 0.25 probability as before.
- **Bubble-sensing with mobility-based bubble maintenance.** This scenario mirrors the previous, but uses mobility-based bubble maintenance which does not require localization for sensing nodes or anchors, but instead uses radio range to define the bubble size and inference of human mobility from accelerometer data [10] to estimate relative location to the

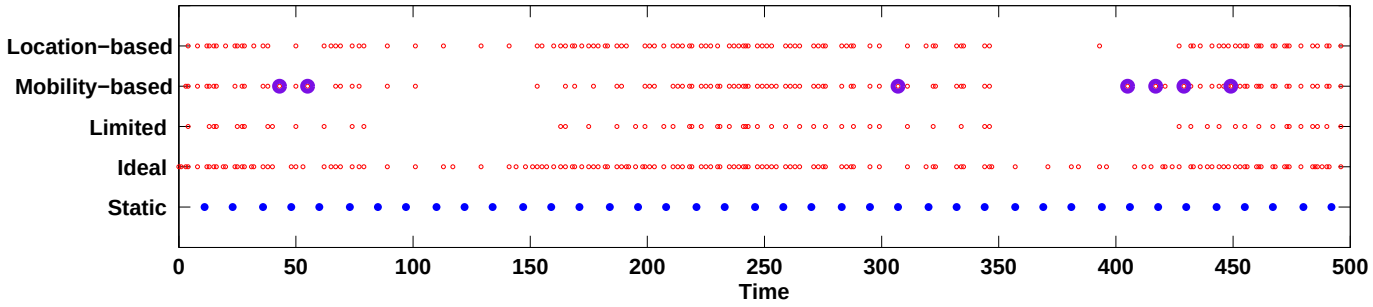

Fig. 3. Sensing coverage over time for each of the five test scenarios described in Section IV-A. The circled points are samples taken outside the bubble due to bubble drift. The bubble sensing cases do a good job of approximating the ideal case, especially for the location-based bubble maintenance case.

bubble. Mobile nodes entering the bubble location become task carriers with a probability of 0.25 that now includes the probability of having both an available data uplink and localization capability.

The mobility of the human participants is uncontrolled, but clearly plays a dominant role in the sensing coverage achievable with the bubble-sensing system. Similarly, environmental factors impact the noise environment and thus impact the data that are collected by the mobile sensors. To ensure the five schemes are evaluated in the same environment, we implement them all in the same multi-threaded application and collect data for them simultaneously. The data samples are stored locally and forwarded to the bubble server opportunistically when the phone switches to infrastructure mode, for the duration of the experiment. Any remaining data is transferred to a laptop over USB at the end of the experiment. The analysis is done offline in the backend sever.

B. Results

We conduct three trials using 11 mobile phones at different times of the day, including both day and night, to capture natural variations in density and mobility pattern. Trial 1 lasts 1936 seconds during the day-time work hours when people are more stationary; Trial 2 lasts 1752 seconds during the evening-time work hours; trial 3 lasts 1198 seconds during a more mobile period. In some cases, we did not get data from all the mobile phones; some did not enter the bubble, and for others the user profile prohibited them from participating (i.e., emulated by the 0.25 probability for task download). We got data from 9, 8, and 7 for the trials 1, 2, and 3, respectively. Table II shows raw sample counts taken during each of the trials for each of the five schemes. The table also indicates the samples taken outside of the bubble due to drift in the mobility-based scheme.

Figure 3, shows the time distribution of the collected data samples. It is a 500 second snap shot of trial 2. Each dot in this figure represents one sample. The Y axis lists the five schemes we compare. The distribution of all the mobile schemes is not uniform, because the ability to sense is influenced by uncontrolled mobility. For the mobility-based bubble-sensing scheme, the circled dots show where samples are taken outside the bubble due to bubble distortion and drift. In all mobile

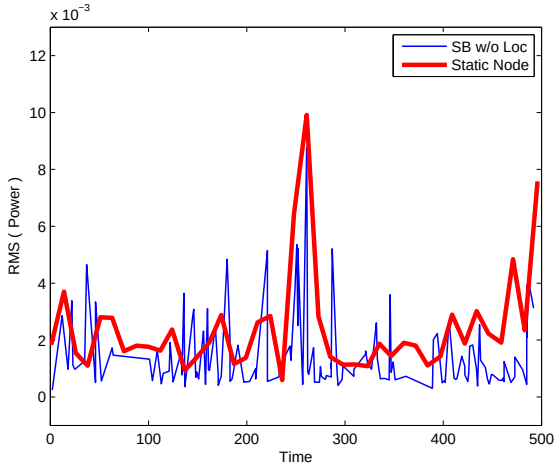

Fig. 4. In terms of data fidelity, the bubble sensing approach provides sound data whose trend follows that of the static sensor. In practice, the required fidelity of the signal captured by the task is application-specific.

schemes, sometime we have dense readings because multiple sensors stay in the bubble, and sometimes there is a gap in the sensor data due to the absence of sensors. In terms of sensing coverage over time, the bubble-sensing schemes give a good approximation of the ideal mobile sensing scenario, especially in the location-based bubble maintenance case. For the mobility-based bubble maintenance base, we see that the percentage of samples taken outside the defined bubble is less than 10%, which we conjecture is an acceptable error given the flexibility the scheme provides in not requiring localization for sensing nodes or bubble anchors. Further, data just outside the bubble may still be of use to the data consumer.

To examine how the data collected by the bubble-sensing system compares with that from the static node, we compute the root mean square (RMS) of the average sound signal amplitude. In Figure 4, we plot the RMS derived from every sound clip recorded by the two different schemes, the static node (thick red) and bubble sensing with mobility-based bubble maintenance (thin blue). The bubble-sensing curve contains more data points (140) than the static curve (41), reflecting the mobile nodes that opportunistically sample over

the 500 second window, as opposed to the single periodic static sensor. While the two curves share general trends, they do not match exactly. There are two main factors contributing to this phenomenon, i.e., mobility and context. The static node remains stationary 3 meters from the sound source, while the mobile nodes move in and out of the audio range of the source, affecting the volume of the samples. Another factor affecting the volume is the sampling context. Users carry the cell phone in their pocket and the pant material serves as a kind of muffler. Also, the orientation of the microphone matters. However, the sampled data does match the general sound situation in the target region, which may be good enough to support applications when static sensor deployments are not present. Thus, bubble-sensing provides the flexibility of personalizable sensing regions, but sacrifices some signal fidelity.

V. Related Work

While the mobile phone is ubiquitous, and the discussion of a mobile phone as a sensing device has some history [8] [5] [2] [1], no large-scale mobile cell phone sensor networks have yet been deployed in practice. In the last year, the smart phone market has grow rapidly (e.g., Nokia N95, Apple iPhone, Goggle gPhone), leading to a great research opportunity. In this paper we present our first attempt to build a mobile cell phone network.

Use of information locality to achieve efficient, scalable sensor networking is a hot topic. Ratnasamy, *et al* discuss the use of data centric storage to reduce the transmission overhead [11]. The authors of [9] use a publish/subscribe mechanism to opportunistically disseminate information within a specified geographical area using a vehicular networks. In contrast, in our work we focus on the locality of the sensing task, and discuss how to fulfill the sensing task on top of a cloud of human-carried smart phone-based sensors in the urban sensing context.

VI. Conclusion

We presented an approach to support persistent location-specific task in a wireless sensor network composed of mobile phones. Mobile sensor nodes collaborate and share sensing and communication resources with each other in a cooperative sensing environment. We describe the virtual roles nodes can assume in support of bubble-sensing, including the required local and backend communication. We discussed the limitations, available options and our design decisions in the implementation of a mobile phone-based sensing system. We demonstrated the feasibility of our scheme via a real test bed experiment using people carrying mobile phones.

Acknowledgment

This work is supported in part by Intel Corp., Nokia, NSF NCS-0631289, and the Institute for Security Technology Studies (ISTS) at Dartmouth College. ISTS support is provided by the U.S. Department of Homeland Security under Grant Award Number 2006-CS-001-000001. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. Department of Homeland Security.

References

[1] T. Abdelzaher, Y. Anokwa, P. Boda, J. Burke, D. Estrin, L. Guibas, A. Kansal, S. Madden, J. Reich. Mobiscopes for Human Spaces. *IEEE Pervasive Computing*, vol. 6, no. 2, pp. 20-29, Apr-Jun, 2007.

[2] J. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, M. B. Srivastava. Partcipatory Sensing. In *Proc. of 1st Workshop on Wireless Sensor Web (WSW'06)*, pp. 1–5, Boulder, October 31, 2006.

[3] A. T. Campbell, S. B. Eisenman, N. D. Lane, E. Miluzzo and R. A. Peterson. People-Centric Urban Sensing (Invited Paper). In *Proc. of 2nd ACM/IEEE Int'l Conf. on Wireless Internet*, Boston, Aug 2-5, 2006.

[4] K. Chang, N. Yau, M. Hansen, and D. Estrin. SensorBase.org A Centralized Repository to Slog Sensor Network Data. In *Proc. of the Int'l Conf. on Distributed Computing in Sensor Networks/Euro-American Workshop on Middleware for Sensor Networks*, San Francisco, Jun 2006.

[5] N. Eagle and A. Pentland. Reality Mining: Sensing Complex Social Systems. In *Journal of Personal and Ubiquitous Computing*, Jun 2005.

[6] S. B. Eisenman, E. Miluzzo, N. D. Lane, R. A. Peterson, G.-S. Ahn, and A. T. Campbell. The BikeNet Mobile Sensing System for Cyclist Experience Mapping. In *Proc. of 5th ACM Conf. on Embedded Networked Sensor Systems*, Sydney, Nov 6-9, 2007.

[7] S. B. Eisenman and A. T. Campbell. "SkiScape Sensing". In *Proc. of ACM 4th Intl Conf. Embedded Networked Sensor Systems*, 2006.

[8] A. Kansal, M. Goraczko, and F. Zhao. Building a sensor network of mobile phones. In *Proc. of 6th Int'l Conf. on Information Processing in Sensor Networks*, Cambridge, Apr 25-27, 2007.

[9] I. Leontiadis and C. Mascolo. Opportunistic spatio-temporal dissemination system for vehicular networks. In *Proc. of 1st Int'l Mobisys Workshop on Mobile Opportunistic Networking*, San Juan, Jun 11 2007).

[10] J. Lester, T. Choudhury, and G. Borriello. A Practical Approach to Recognizing Physical Activities. In *Proc. of Pervasive*, Dublin, May 2006.

[11] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker. GHT: A Geographic Hash Table for Data-Centric Storage in SensorNets. In *Proc. of 1st ACM Int'l Workshop on Wireless Sensor Networks and Applications*, Atlanta, Sep 2002.

[12] K. Romer, C. Frank, P. Jose Marron, and C. Becker. Generic role assignment for wireless sensor networks. In *Proc. of 11th ACM SIGOPS European Workshop*, pp 7–12, Leuven, Sep 2004.

[13] CPUMonitor 1.10 for S60v3. http://www.nokiapower.com/index.php?showtopic=7542.

[14] G. M. Djuknic and R. E. Richton. Geolocation and Assisted GPS. *IEEE Computer*, Vol. 34, No. 2, pp. 123-125, Feb, 2001.

[15] A. LaMarca, Y. Chawathe, S. Consolvo, J. Hightower, I. E. Smith, J. Scott, T. Sohn, J. Howard, J. Hughes, F. Potter, J. Tabert, P. Powledge, G. Borriello and B. N. Schilit. Place Lab: Device Positioning Using Radio Beacons in the Wild. In Proc. of Pervasive, pp. 116-133, 2005.