Journal of the Indian Institute of Science
A Multidisciplinary Reviews Journal
ISSN: 0970-4140 Coden-JIISAD

REVIEWS

Empowering Feature Phones to Build Smart Mobile Networked Systems

Kuldeep Yadav and Vinayak Naik

Abstract | In last two decades, increase in mobile devices coupled with subscriptions have enabled large proliferation of mobile systems and applications. Apart from traditional services such as voice calls, SMSes, today's mobile phones enable array of other services i.e. location-based services, multimedia, education, citizen information services for the consumers. However, developing mobile system which can work on all the mobile devices is still a challenge to solve due to large heterogeneity among mobile devices and network. Most of mobile systems are built for smartphones, which constitute of only small percentage of all mobile devices, rest of devices are feature phones, which have limited capability in terms of computation or communication.

We need to take special design considerations while building mobile systems for feature phones. In this paper, we present three of such mobile systems, which are carefully designed for mobile phones in resource-constrained environments. First, CBS-based localization provides a positioning method for feature phones, which works without war-driving. Second, a system called *Unity* that enables collaborative download among co-located mobile users and third, a participatory sensing system *Human Sensors*, which enables human-in-loop sensing of city problems using a mobile phone.

1 Introduction

The total number of mobile phone subscriptions were nearly 6.8 billion in early 2013 covering more than 95% of world population.[16] At the same time, mobile devices shipments continue to rise every year. According to Gartner, 1.75 billion handsets were sold in 2012 and it is predicted that in 2013, shipments will grow to 1.90 billion.[15] Similarly, mobile networks have evolved from 1*G* to 4*G* in last two decades. Increase in mobile devices coupled with subscriptions have enabled large proliferation of mobile systems/applications. Apart from traditional services such as voice calls, SMSes, today's mobile phones enable array of other services such as location-based services, multimedia, education, mobile payment, social media, gaming, and citizen information services for the consumers.

Till now, most of the research has focussed on building services for smartphones which are equipped with powerful set of sensors such as compass, accelerometer, GPS, WiFi etc., and also, have access to high speed data connection (LTE/HSPA/HSDPA). However, there are large number of phones which do not have access to these sensors popularly called as feature phones. As an estimate from Gartner, number of feature phones are nearly 75% percent of the total 4.3 billion phones in 2012. Many of these services do not work on feature phones due to lack of sufficient hardware and limited bandwidth data connection (GPRS/EDGE).[48] We need to take special design considerations for designing mobile systems for feature phones. In this paper, we describe design and development of mobile systems, which will empower feature phones in

Indraprastha Institute of Information Technology, Delhi (IIIT-D), New Delhi, India.
kuldeep@iiitd.ac.in
naik@iiitd.ac.in

three aspects, namely, **Localization, Communication**, and **Sensing**.

Current user location has been an integral part of user context and an enabler of many services like navigation, activity recognition, local business search, and friend finder services. Interestingly, all context aware services do not require same level of accuracy for current location. For instance, navigation applications require high level of accuracy (~10 meter) whereas if one has to share location with online social networks, even location accuracy of hundreds of meter will suffice. There are mainly three localization technologies which can be used to get user's current location on mobile phone i.e. Global Positioning System (GPS),[28,38] WiFi-based[19] and GSM-based.[24,40]

Among these technologies, GPS is highly accurate (~10–100 meter) and works with the help of at least four satellites. However, it consumes high energy, requires special hardware, and only works outdoors.[50] It is predicted that for the next five years, over 50% of the phones will not have GPS.[43] WiFi-based localization requires an already established WiFi infrastructure and a perceptual map of wireless APs identifiers with respective signal strengths.[26] The process of creating perceptual map is called as war-driving. WiFi-infrastructure does not exist at a city-scale in many countries and war-driving is a cost intensive process.[47] Continuous use of GPS and WiFi drains mobile phone battery very quickly,[30] many studies have found that power consumption is one of the biggest barrier in large scale adoption of location-based services.[33] For the class of applications that do not require fine grained location accuracy, GSM-based localization is better suited due to its wide availability and low power consumption. In fact, for low-end phones (without GPS/Wi-Fi capability) this is best suited.[43]

As per recent statistics, number of 3*G* subscribers in China are only 14% of the total 1 billion cellular subscribers,[1] while in India, it is only 2% of over 893.8 million subscribers.[2] Low penetration of 3*G*/4*G* networks is attributed to higher setup cost, limited number of supporting handsets, and expensive data plans. As a result, many people still use 2*G* based data connection which is widely deployed and accessible on most of the phones. In countries like India and Egypt, over 50% of users have access to Internet from mobile phones only.[14] Data consumption rate of the mobile phones is increasing every day due to requirements posed by mobile applications/services such as multimedia. Recent Opera report shows[a] that, mobile users download content (mostly multimedia) from the Internet using 2*G* network and the top handsets used were found to be feature phones.

To enable faster downloading of content (workload), we present *Unity*, a system that enables collaboration between multiple co-located phones (peers). In *Unity*, co-located peers participate to individually download different parts of the desired workload and then share their downloaded part with each other such that everyone gets the complete content at the end. Mobile phone users are typically expected to be part of several social gatherings during the day at different places i.e. home, workplace, and even while commuting.[44] The key idea of *Unity* is to use these social meetings to provide a collaborative environment with a usable interface to enable faster downloads of content desired by all (or most) of the participating peers. Almost all the phones, including feature phones, have one or more small range radio technologies, such as WiFi, Bluetooth, NFC. *Unity* leverages one of the available short range technologies for coordinating and local sharing of workload parts amongst peers while each of the part is downloaded by individual peer from the Internet using cellular connection.

Many developed countries have city-wide deployment of sensing infrastructure to collect data about day-to-day city events, the collected data is then analyzed online or offline to take a prompt action based on those events. For instance, USA have deployed traffic sensors across major highways to monitor the health of roads and to detect timely events such as traffic congestion. Data collected from these sensors is useful to make broad city development decisions. This kind of sensing infrastructure does not exist or have limited coverage in many countries due to lack of resources, cost, and bigger scale of deployment. Being an integrated computing, storage, and communication device, mobile phones can be efficiently used for sensing tasks.[34] There has been several efforts to use smartphone sensors for variety of purposes such as to estimate pollution exposure,[36] pothole detection[27] and traffic conditions.[35] All of these work expects people to participate, collect appropriate sensor data using their smartphones and contribute it for a common purpose.[23] We have designed a participatory sensing system to collect information about city events with the help of citizens while offering different submission interfaces i.e Android-based mobile application, SMSes, and a web based tool. Multimodal interface of the system allows citizens to sense and submit events using their preferred mode which can minimize cost. Findings from processed events data can be

[a] Opera Mini is a widely used browser in mobiles.

used to multiple purposes i.e. citizens information, city planning etc.

The paper is organized as follows, Section 2 presents design goals which needs to be take care of, while building system for feature phones. Section 3 describes a GSM-based localization system which works on feature phoneswithout any extra infrastructure. Section 4 presents a system *Unity* that enables collaboration between multiple co-located peers to download a mutually desired content using low bandwidth EDGE connections. Finally, we have described a city-scale sensing system *Human Sensors* in Section 5 and discuss our findings and experiences in Section 6.

2 Design Goals

Feature phones have limited capabilities as compared to smart phones, system designers have to take special consideration while building system for these phones. Following are some of design goals which needs to be taken care of while designing such systems.

- **Low Cost**: Mobile users especially in developing countries are cost-constrained. A mobile system should perform its functions with minimal cost. For instance, most of participatory sensing systems are voluntary,[34] they should be designed to minimize user cost as much as possible.
- **Low Energy**: One of the main concern with mobile devices is limited battery. A mobile system design should try to maximize battery life of the device. For example, services which uses network connection very frequently tend to drain battery very quickly. There have been approaches which leverage delay tolerance of web traffic to schedule them in same data connection session which can potentially minimize energy.[21,20] Similarly, low RSSI results in less throughput and more energy consumption.[39] Hence, a data connection should be initiated only when RSSI is good.
- **Limited Hardware and Programming Support**: A mobile system has to work with limited hardware and programming support to provide smartphone like service to feature phones. For instance, GPS and WiFi chip does not exist on low cost phones.[43] A mobile system should use GSM-based location interfaces as much as possible which are available on most of the phones.[47]
- **Utilizing Available Resources**: A mobile system should be designed to utilize available resources to fullest. Almost all the phones, including feature phones, have one or more small range radio technologies, such as WiFi, Bluetooth, NFC. Short range radio technologies can be used intelligently to share content or resources locally.[46]

While above design goals are described keeping feature phones and developing countries as a focus, they are general in nature and hold true for any community or mobile phones.

3 Localization

GSM-based localization is most preferred way of accessing user location for feature phones due to its wide availability and minimal energy consumption. However, current approaches have limitations which are described in next subsection.

3.1 *Background*

Prior work related to GSM-based localization can be divided into two categories: (A) Cell ID based Approaches and (B) RSSI fingerprinting based approaches.

3.1.1 Cell ID-based approaches: In this approach, Cell IDs are fetched using phone APIs, and looked up in an existing war-driving based database to provide localization. To the best of our knowledge, none of the mobile phone operators reveal exact location of the Cell towers. Hence, using crowd sourcing/war driving data, cell tower location is approximated, which could be several hundred meter away from its actual location. According to GSM standards, a phone can receive signals from seven different Cell towers.[25] If there are multiple visible Cell IDs, the approaches compute some function, e.g. centroid, of all the geo-coordinates (latitude and longitude) obtained from the database. Our experience supported by other prior work[38,37] show that for several phones (including Nokia *S*40, *S*60 phones, Samsung Android phones) provide access to only one Cell ID to which the phone is currently connected. This significantly reduces accuracy of the localization as compared to the accuracy that had been obtained with access to seven Cell IDs.

Google Mobile Maps'(GMM) My Location[8] application works on a single Cell ID-based approach, where it provides a median localization error of 656.37 meters for a rural area and 503.89 meters for an urban area.[31] The localization error depends on density of cell towers. Due to high density of Cell towers, this method provides better accuracy in urban areas as compared to rural area. However, it is hard to get a comprehensive database of Cell IDs. There are some proprietary databases, such as one used by GMM, which are not publicly shared.

There exist open source initiatives, e.g. OpenCellID[17] and Cell Spotting,[5] that build their database using crowd-sourcing. To check the coverage of open source cell ID databases, we selected two widely used operators in New Delhi. We call them X and Y for anonymity. On our self collected dataset of Cell IDs for operator X, we observed that out of 252 cell IDs, OpenCellID contained only 65. For operator Y, the number was only 21 out of 164 as shown in Table 1. We cannot find out comprehensiveness of the GMM as it is not publicly available. However, given low penetration of Android phones in rural India, we postulate that the database will be underpopulated. Crowd-sourcing for building cell ID database seems to be ineffective due to (A) lack of incentives as people need to incur airtime charges for contributing to the databases and (B) lack of GPS-enabled phones in developing countries.

3.1.2 RSSI fingerprinting-based approaches: In this approach, RSSI (Received Signal Strength Indication) is collected along with Cell IDs during war-driving and a perceptual map is built similar to WiFi-based approaches. Typically, a fingerprint constitutes of Cell IDs, their associated RSSI, and GPS coordinates that are represented in a vector form. During the localization phase, Cell ID(s) and associated RSSI are compared with stored vector space of fingerprints using KNN (K Nearest Neighbor) to estimate user's location. KNN uses euclidean distance in RSSI space as a metric to find closest stored fingerprint.[25] This approach gives better accuracy than the Cell ID-based approaches since granularity of stored information is more. However, it requires more storage and computational capabilities as well as more efforts are needed during war-driving.

Continuous war-driving effort is required in this approach because RSSI keeps on fluctuating due to changes in physical environment. It works good when there is a visibility of seven cell towers and their respective RSSIs. Recent results demonstrate that RSSI measure from single cell tower is not a good measure to calculate movement.[38] We conducted our own study to find out whether RSSI is a good metric for localization. RSSI difference is the absolute change in the RSSI, for a given Cell ID, when user moves from one location to another. In our database, we had 24064 unique RSSI difference values from 410 unique cell IDs. We plot maximum, minimum, and average distances for each RSSI difference. As seen in Figure 1, the average difference is almost constant for RSSI difference ranging from 1 to 9 dBm. This concludes that RSSI is not a good measure for GSM-based localization as one observes similar RSSI values between two points with large physical distance between them.

This concludes that current GSM-based localization approaches have practical limitation such as limited access to APIs, need of war-driving etc. In the next section, we will describe CBS-based localization approach which do not require war-driving effort and works with APIs provided by major phone platforms.

3.2 *Architecture*

The CBS messages are broadcast by Cell towers to all the phones in its range.[13] CBS is defined in the phase II of GSM standard 3.49.[4] The users need not pay airtime charges to receive CBS messages, even while roaming outside of their home area. The CBS messages are commonly used to broadcast information about weather forecast, landmarks/area names, news, announcement by governments, etc. All this information can be broadcasted simultaneously on different channels. A cell tower typically broadcasts the locality/landmark name, where it is located. Channel 50 is reserved for broadcasting location/area names. Most of the phones come with built-in APIs to capture CBS messages. Figure 2 shows architecture of a working instance of CBS-based approach. The data flows as depicted

Table 1: Success rate of Open Cell ID (most extensive open source database of cell IDs) on our dataset collected in New Delhi region.

Operator	No of cell IDs	Found on OpenCellID	%
X	252	65	31%
Y	164	21	13%

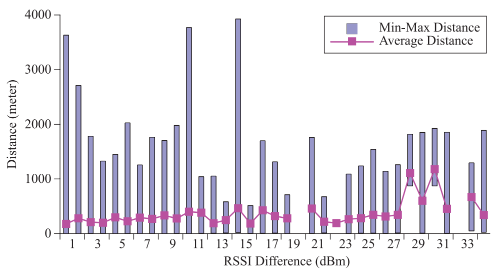


Figure 1: In whole dataset, Min-Max bars representing minimum and maximum distance between two position into same cell ID. For instance, for a difference of 14 dBm in RSSI, distance between those points can range from 0 to 4000 meter.

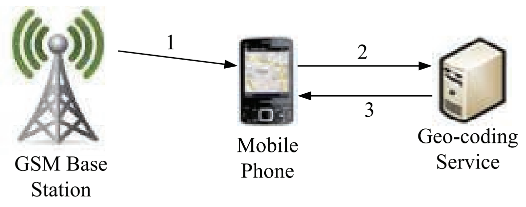


Figure 2: Architecture of CBS based Localization System.

by numbered arrows in the Figure 2 correspond to the following:

1. GSM base station broadcasts CBS messages, each containing a CBS string mentioning location name or advertisement. The messages are received by a listener application running on the phone. The listener application at the phone continuously listens at the port 50 for incoming messages.
2. After receiving a message, mobile application automatically figures out whether it contains location or advertisement. If the message content is a location, then the phone checks for its corresponding geo-coordinates in its local cache. If it is not available, the application makes a request to geo-coding service.
3. Geo-coding service estimates the geo-coordinates of the queried location name using a combination of techniques described in Section 3.3.2 and return them to the phone. The application at the phone adds it to local cache of the phone. Geo-coding service is likely to get request from many phones, using that it builds a cache of all location names with their geo-coordinates. Phone application can download this global cache proactively to avoid frequent requests to the cloud.

Above described approach is the most basic way of estimating a user's location using CBS messages and called as baseline approach. Baseline approach is identical to Cell ID approach described in Section 3.1. To the best of our knowledge, this is the first attempt to use CBS messages for localization. We could not find any publicly available dataset of CBS messages. To characterize the accuracy of CBS-based localization approach, we collected CBS messages for two different operators X and Y along with GPS coordinates in urban settings of New Delhi, India. Rate of reception of CBS messages depends on the speed of the user, we have characterized our collected traces in two categories i.e. walking and traveling as shown in Figure 2. Our detailed data collection methodology is described in our earlier work.[47] Further, we have also procured a CBS message dataset from Nokia which is collected as part of pilot deployment of Nokia Nearby application. This dataset has about 29*K* records with 469 unique CBS location names.

We analyze the data from both the datasets in the next section and list out challenges in using CBS messages for localization.

3.3 *Challenges*

Data from our pilot study brought forth several non-trivial challenges which need to be solved before CBS-based localization can be realized in real-world setting. In this section, we describe all challenges and presents our approaches to solve them.[47,49]

3.3.1 Filtering of advertisement messages:

CBS messages contain advertisements in addition to location names. It is essential to filter out these advertisements. By looking at the data, we have made the following two observations which can distinguish between a location name and an advertisement.

1. Advertisements contain common patterns such as special characters ('*','#','%','@') or continuous digits like ('578785') which are unlikely to be present in genuine location names.
2. Advertisements contains operator name or some other advertisement specific words such as "Cricket", "Free", "Ringtone", which are hard to find in genuine location names.

Using these two discriminators, we designed a regular expression which can filter all the advertisements at the phone itself.[49] These observations hold true for both the operators' data. We got 100% accuracy in filtering advertisements when the regular expression was applied off-line to 33279 CBS messages in our dataset. Interestingly, we found that number of advertisements differ among operators X and Y as shown in Table 3, operator Y had about 61% advertisement CBS messages whereas operator X had about 46%.

Table 2: Number of travelling and walking traces in CBS dataset across two different operators X and Y.

State	X	Y	Combined (X + Y)	Avg Duration (Minutes)
Travelling	27	12	10	46
Walking	12	7	7	65

Table 3: Percentage of advertisement CBS messages in both the datasets collected for operator X and Y.

Operator	Total CBS messages	Advertisements (%)
X	32106	46%
Y	1173	60.53%

3.3.2 Geo-coding of location names: As described in Figure 2, CBS location messages need to be converted into corresponding geo-coordinates using a geo-coding service. Among all the on-line maps services, we found Google Maps to be most effective in geo-coding CBS location names because its coverage is much higher than other map providers specifically in the areas where we had collected data. We have used Google Maps's geo-coding APIs[7] for all our experiments. We found that more than 30% of location names can not be geo-coded directly by Google APIs. Primary reasons of geo-coding failures are described as following:

1. Location names may exist differently (in the geo-coding service), e.g. there could be a spelling difference, use of short hand abbreviations etc. For example, 'Matiyala' and 'Matyala', 'Uttam Nagar' and 'Uttam Ngr', 'Dwarka Sec-3' and 'Sec-3 Dwarka'.
2. Some locations have two different names i.e. one could be colloquial which local people use and other one is official name which exist on Maps. For example, 'Kakrola Mor' and 'Dwarka Mor' etc.
3. Some location names do not exist completely on maps as coverage of these services in developing regions are still limited and there is no publicly available extensive GIS database by government agencies too.

To increase geo-coding success rate, we propose following geo-coding framework which uses combination of several techniques described as follows:

1. **Direct Geo-coding:** The location name which need to be geo-coded are directly sent to Google Maps's geo-coding APIs. If it exists on maps, APIs will return corresponding latitude and longitude information. Otherwise, it will return a geo-coding failure.
2. **Pre-processing Location Names:** If there is a geo-coding failure in the previous step then pre-processing is done on location name to remove ambiguity, if any. Presence of irregular or no-space, extra characters, and short hand abbreviations in some location names makes it difficult for direct geo-coding to find their coordinates. To resolve the ambiguity present in location names, we do a pre-processing of location names before sending them to the geo-coding service. Pre-processing algorithm apply following steps to sanitize the CBS location names:
   a. Replace special character such as '-' with a space. For example, location names such as 'Dwarka Sec-02', 'Dwarka Sec-2' and 'Sec-2-Dwarka' are converted to 'Dwarka Sec 02', 'Dwarka Sec 2' and 'Sec 2 Dwarka' respectively.
   b. Numerical characters in the location name are separated out from surrounding text characters e.g. converting 'Dwarka Sec2' to 'Dwarka Sec 2'.
   c. After fixing special characters and numerical characters in location name, our pre-processing algorithm search for popular abbreviations in location names like 'NGR', 'SEC', 'VHR' etc., and replace them with its full form like 'NGR' for 'Nagar' with the help of a dictionary. We have populated this dictionary from the location names using a semi-automated process.
   d. Similar to the above step, stop words such as 'Nagar', 'Garden' are searched into a location name and a space is inserted before every occurrence of a stop word. For example, 'Rajourigarden' will be converted into 'Rajouri Garden'.

   After sanitizing location names using above listed steps, direct geo-coding is used to convert give location names into coordinates. If geo-coding is successful, process is stopped here otherwise, it will move to next step.
3. **Using Crowd sourced POIs Information:** For the location names that are completely missing from digital maps or exist with different name(s), we take help of point of interest (POIs)/businesses data which are crowd sourced by Google Maps. We use Google Places API[9] to fetch POIs information about a specific location name. Many of CBS location names are colloquial, they were present in this crowd sourced location data in some form or other, but not on Google Maps. As, crowd-sourced data has its own challenges in terms of accuracy, noise etc, which need to be solved before using this data for estimating a CBS location's geo-coordinates. Following is step by step description of our algorithm which

estimates coordinates of a given location using this POIs data.

Step 1: Google Places API requires input of base coordinates, queried location name and it searches POIs around the base coordinates which contains queried location name in their addresses. We use the geo-coordinates of immediate previously received and geo-coded location name from the CBS trace, as base coordinates. If the queried CBS location name has space, we break it into different set of location names and query Google Places API multiple times. For instance, 'Palam Railway Junction' will have three different values for queried location name i.e. ['Palam', 'Palam Railway', 'Palam Railway Junction']. The idea behind creating multiple names/queries is to maximize POI addresses because many times, a part of location name may miss from the actual POI address. For instance, many POIs may contain name 'Palam' because it is a more popular location entity than 'Palam Railway Junction'. Here is one sample query to Google Places API.[10]

Step 2: Google Places API returns a different set of POI addresses with every different value of queried location name. The results are aggregated from all the queries and a single set of POI addresses is maintained with their corresponding geo-coordinates. After aggregation, we apply levenshtein string edit distance to rank POI addresses based on their lexical similarity to the actual CBS location name and select top-K addresses.

Step 3: After selecting top- K POI addresses, we compute an average of their geo-coordinate. The resultant coordinates will be the estimated coordinates for a given CBS location name.

4. **Geocoding Framework Evaluation:** From both the datasets, we had total 572 unique location names. In our data collection, we have collected GPS coordinates too, with the CBS location names. We use GPS coordinates as a ground truth to evaluate accuracy of geo-coding framework as well as individual techniques. From our empirical evaluation, we have found value of K equal to 10. Our geo-coding framework successfully geo-coded nearly 92% of total 572 location names as shown in Table 4. Our pre-processing algorithms and use of crowd sourced information jointly increased the geo-coding success rate by 26.39%. Pre-processing location names increased the geo-coding success rate by only 4.54%. In our earlier work,[47] pre-processing of location names increased geo-coding success rate by about 15% on a relatively smaller data-set of 143 location names. From our close observation, we have found that Google Maps have improved over time and they are already doing some pre-processing of location names which we have done earlier in.[47]

Table 4: Success percentage of different steps in geo-coding.

Geocoding frameworks	Successfully geocoded	Successful (%)
Direct Geocoding	376	65.73%
Pre-processing + Direct Geocoding	26	4.54%
Using Crowdsourced POIs information	125	21.85%
Total	527	92.13%

We believe that a common algorithm that can work for geo-coding of all the location names is very hard to achieve due to non-standard nomenclature for CBS messages and poor GIS database (specially in developing countries). However, it is still a one-time task to geo-code the names which are not automatically geo-coded by any service and requires much less effort than the war-driving used by other GSM-based localization approaches.

3.4 *Inaccuracy of CBS location messages*

In Cell ID-based localization approach (described in Section 3.1), accuracy depends on the richness of the perceptual map (Cell ID database) that is created using war-driving. Similarly, accuracy of CBS-based localization approach will depend on quality of location names that we receive as CBS messages. To understand the quantitative estimate of the error in CBS messages, we have computed the localization error which is essentially distance between geo-coordinates of CBS location and associated GPS coordinates. If there are multiple GPS coordinates associated with a CBS location in our dataset, we took an average of all those GPS coordinates and then computed the distance. Figure 3 shows a bar graph of distribution of error in terms of percentage of the location names. Out of 527 location names, about 8% of the names could not be geo-coded, we have ignored such names while plotting Figure 3. For about 58% of the names, which were successfully geo-coded, localization error was more than 500 meters. Typically, CBS messages have inaccuracies, such as

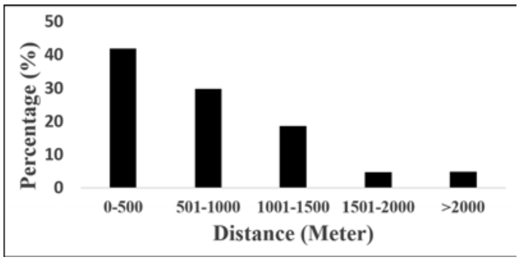


Figure 3: Distribution of localization error for all the location names. For 58% of the location names, error is more than 500 meters.

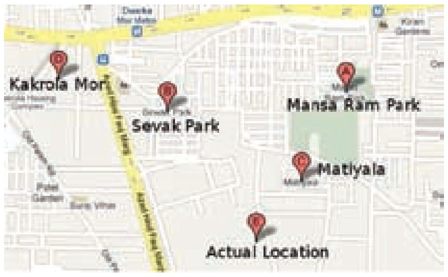


Figure 4: Snapshot of received CBS location messages at a given location. Marker *E* depicts the current location of the phone and markers *A–D* presents the four CBS location messages received during phone's stay at *E*.

errors introduced by geo-coding services, presence of generic CBS location names, and association of same CBS names with multiple towers.[47] These inaccuracies impact the localization accuracy and to reduce the impact, we design two algorithms described in next section.

3.5 *Algorithms to improve localization accuracy*

CBS-based localization approach is primarily aimed for feature phones. Primary design goal for CBS-based localization algorithms is to strike a balance between reasonable accuracy and being low-cost in terms of computation, communication, and data storage. Baseline approach takes the most recently received CBS message's geo-coordinates to approximate the location of the user. Baseline approach does not always give good results due to inherent errors described in Section 3.4. A key insight towards reducing the impact of these errors is that we are not taking into account history of the locations visited by the mobile users in the recent past.

To account for location history, we form a vector of location names received in the past. When the user is stationary, the phone often receives multiple distinct location names as it can associate with different cell towers that are in geographic proximity at different time instances.[22] Figure 4 presents a real example from our dataset which shows reception of four different CBS messages even when user stays at same place. As shown in Figure 4, CBS messages sometimes may include locations that, in reality are far away from user's current location. However, the frequency of such location names may be smaller than frequency of location names that are in close proximity to the current location. We hypothesize that this frequency difference is a factor of distance between Cell Tower and the user. Therefore, a weighted average based approach where the weight given to each location name is dependent on the frequency of received messages with the corresponding location name (within fixed time window) will intuitively work well for improving the localization accuracy. We call this approach *FrequencyWeighted* and it considers all CBS location messages received in a fixed time window duration δ. For instance, if a service running in a mobile phone requires user's current location at time t, *FrequencyWeighted* algorithm takes all the CBS location messages which were received between $(t, t-\delta)$ and compute estimated user location by taking weighted average of their corresponding geo-coordinates. The detailed explanation and pseudocode of this algorithm is given in our prior work.[47]

For a slow moving user, since the conditions are similar to a static user, the *FrequencyWeighted* approach should ideally provide better localization accuracy. However, a fast moving user will probably be in the range of a Cell tower for a short duration and hence will receive a small number of (often only a single) CBS message with the corresponding location. However, it may also happen that the currently received location name corresponds to a location in real world that is ahead on the path of the user while the previously received location name was behind on the path of the user (a typical case when the location name is received immediately on crossing the cell boundary). Therefore, weighted average of the geo-coordinates of received location names with higher weight given to those that are received most recently and exponentially reducing the weights of location names received in the past will intuitively improve the localization accuracy. This approach uses a timeout parameter λ to check if there is a long gap in the reception of a CBS location message, the algorithm forgets past history of messages and starts accumulating new history if there is a timeout. We call this approach *TimeWeighted* and detailed algorithm is described in our earlier work.[47]

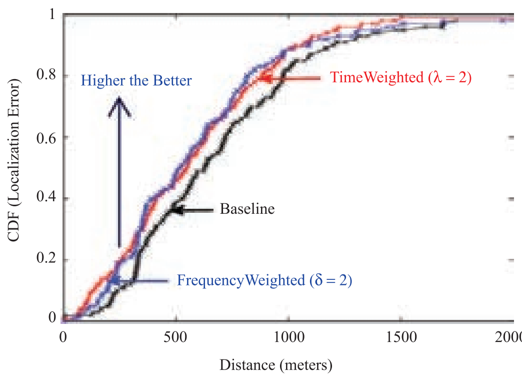
3.6 *Evaluation of the algorithms' accuracy*

We now describe the empirical evaluation of the two algorithms, *FrequencyWeighted* and *TimeWeighted* , explained in the previous section, using our self collected CBS traces dataset. For comparison purpose, we used baseline CBS-based localization approach which is identical to Cell ID based localization approaches (including service providers like Google). We use *localization error* as our evaluation metric. It is the distance between actual location (GPS Coordinates) and estimated location using CBS-based approach. For simplicity purpose, we discuss only one operator's result (referred to as operator Y) in detail and briefly present results for operator X. As hypothesized earlier, the accuracy of the algorithms could depend on the speed of travel. Hence, we evaluate our algorithms on traces for two different modes i.e. static/walking and traveling.

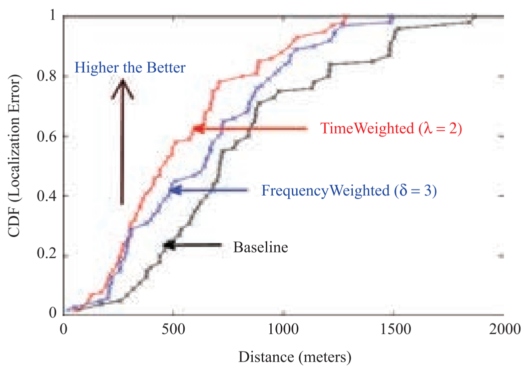
Let us first analyze the effect of varying input parameters on the performance of two algorithms. For *TimeWeighted* algorithm, λ is a timeout parameter, which is necessary to forget old history. Empirically, we found value of λ to be 2 minutes since it gave the least median localization error for all the traveling traces. For *FrequencyWeighted* algorithm, time window to perform weighted average i.e. δ was found to be equal to 2 minutes for traveling traces since it gave the least median localization error. In case of walking traces, we have not found any time-out instance but still kept value of λ to maintain uniformity. We have found value of δ to be equal to 3 minutes. Intuitively, higher value of δ, as compared to the case oftraveling traces, is justified since longer history of CBS location messages will be useful as user is mostly static or walking at slow speed.

Figure 5a compares the Cumulative Distribution Function (CDF) of localization error for *TimeWeighted* and *FrequencyWeighted* algorithm with the baseline approach. Both *TimeWeighted* and *FrequencyWeighted* algorithms perform consistently better than baseline. The improvement in median localization accuracy for *TimeWeighted* and *FrequencyWeighted* over baseline is approximately 12% and 16% respectively. Let us discuss intuition for performance of the two algorithms for traveling case. Typical rate of arrival of CBS message is 1per minute which is consistent across both the operators. With λ fixed to 2 minutes and assuming average speed of traveling trace as $30KM/hr$, if no CBS message is received for 2 minutes, the user has approximately moved by $1KM$ from the location of previously received CBS message. It is therefore better for *TimeWeighted* algorithm to discard the history of CBS messages than to consider them for future calculation of localization. Similarly, with δ fixed to 2 minutes, *FrequencyWeighted* algorithm will onlyconsider CBS messages received within a distance of $1KM$ for calculation of localization, giving weights based on frequency of each CBS message received. This will mostly translate to average of two distinct CBS messages received in the 2 minute interval.

Therefore, in case of traveling trace with correspondingly fixed parameter values, *FrequencyWeighted* algorithm never considers any CBS message outside the 2 minute window while the *TimeWeighted* algorithm gives any message outside the 2 minute window a small weight in case there is no time out in received rate of CBS messages. If there is a timeout happens in *TimeWeighted* , for the first 2 minutes, calculated localization for both of the algorithms will be same. This led to nearly similar performance of both the algorithms in case of travelling traces. For operator X too, both algorithms perform equally good as compared to baseline. The improvement in localization accuracy for *TimeWeighted* and *FrequencyWeighted*



(a) Comparison for travelling traces



(b) Comparison for walking traces

Figure 5: CDF plots for *TimeWeighted* and *FrequencyWeighted* algorithms w.r.t Baseline for operator Y.

over baseline is approximately 10% and 11% respectively, as shown in Table 5.

Figure 5b show the CDF plot where performance of *TimeWeighted* and *FrequencyWeighted* algorithm is compared with baseline for walking traces. Overall *TimeWeighted* and *FrequencyWeighted* algorithms give median accuracy improvement of approximately 35% and 10% respectively over the baseline approach. Earlier, we had hypothesized *FrequencyWeighted* algorithm to provide higher localization accuracy than *TimeWeighted* algorithm for walking traces (as discussed in Section 3.5). However, empirical study showed otherwise. Close observation of the collected data revealed that the walking traces contained a lot of location names, that were farther located, 1200–1500 meters from phone's actual location. This noise, particularly, gets added by the geo-coding service and presence of distant location names, which are among the challenges mentioned in Section 3.3. Effect of this noise can also be seen in terms of higher baseline error for walking traces (712.94 meter) as compared to traveling traces (621.4 meters). In case of *TimeWeighted* algorithm, when such a CBS message with distant location name is received most recently, the calculated location is inaccurate. However, as the time progresses the weight of the CBS message with distant location is reduced, correspondingly resulting inaccuracy is reduced in estimated location as well.

We conclude that our initial assumption that fast and slow motion patterns would demand different approaches for improved localization was empirically found incorrect on our collected data. As shown here, *TimeWeighted* algorithm that was hypothesized to handle fastmotion suffices for slow motion as well since it tolerates the noise added by the distant CBS location names for real data. However, we believe that the localization accuracy may vary slightly across different environments. For operator X, baseline accuracy was good due to good quality of landmarks. Improvement in localization accuracy for *TimeWeighted* and *FrequencyWeighted* over baseline is approximately 18% and 17% respectively, as shown in Table 5.

Table 5: For operator X, Median localization error (in meters) comparison of *TimeWeighted* and *FrequencyWeighted* algorithms with baseline for walking and traveling traces.

Traces	Baseline	*TimeWeighted*	*FrequencyWeighted*
Travelling	688.2	618.29	615.14
Walking	466.69	382.8	386.56

If we don't have ground truth of whether the person is moving slow or fast, we cannot combine the algorithms. However, if we have a way of finding that out, for example using an accelerometer, then the two algorithms can be combined into one to give optimal accuracy. We have used *TimeWeighted* algorithm in designing multimodal approaches, which combines CBS-based localization with Cell ID and GPS to enhance accuracy. Our empirical evaluation shows that combination of Cell ID + CBS can improve the median localization accuracy by up to 40% while Cell ID + GPS can improve the localization accuracy by 51%.

We implement both *FrequencyWeighted* and *TimeWeighted* algorithms as a part of CBS service, which runs on the phone. CBS service continuously listen to incoming CBS messages and stores them in a location vector with their geo-coordinates. Whenever any application needs current location of the user, the service takes location vector as an input and returns calculated coordinates.

4 Communication

Many mobile phone users across the world still use 2*G* based data connection to download large files from Internet. Most advanced 2*G* technology (EDGE) can provide download throughput of up to 48 KBps. We performed an experiment to measure the throughput of 2*G* data connection in wild by repeatedly downloading a MP3 song of about 5 MB size on five different phones, used by volunteers for a week. The median download throughput achieved by the EDGE network across two operators was about 18 KBps while the variation in throughput was from 4 KBps to 28 KBps. We also observed many instances of failed downloads—approx. 22% downloads failing for Operator A and approx. 42% for Operator B. Low throughput and failed downloads are primarily due to two major reasons—variable wireless conditions (low RSSI) and variable load on cellular networks. As a result downloading content (especially multimedia) on EDGE results in several limitations—1) Higher time to download; 2) Excessive power consumption due to low throughput (cellular radio is switched ON irrespective of data speed;[39,41] and 3) Poor user experience due to frequent failed downloads.

There is a lack of appropriate systems that can assist users while downloading content in limited bandwidth conditions offered by 2*G*. It has been observed that co-located people have similar interests w.r.t. downloading of the content. For instance, Figure 6 shows media overlap among 38 mobile phone users in a publicly available dataset. Mobile phone

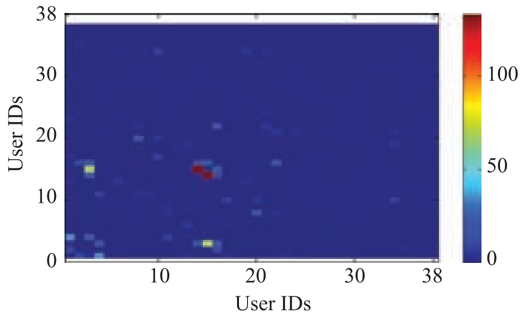


Figure 6: Media file overlap among 38 users in publicly available Nokia MDC dataset. User ID 14 and 15 have 133 common media files where as total 47 user pairs have atleast one common media file among them.

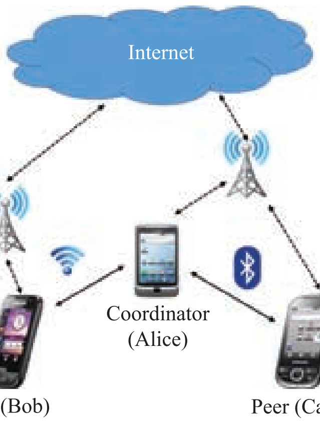


Figure 7: Design of proposed system *Unity*, a group of mobile phones users collaborates with each other to download a single workload which are of interest to all of them.

users are typically expected to be part of several social gatherings during the day at different places i.e. home, workplace, and even while commuting.[44] The key idea of *Unity* is to use these social meetings to provide a collaborative environment with a usable interface to enable faster downloads of content desired by all (or most) of the participating peers.

4.1 *Architecture*

System architecture of *Unity* is guided by various capabilities of commonly available phones. *Unity* works in two different modes for the participating phones i.e. a phone can either act as a coordinator or a peer. The coordinator initiates the download, recruits phones in the geographical vicinity that are willing to participate in the download process, and coordinates all the communication with the participating peers. The peer connects to the coordinator, downloads a part of the desired workload and shares it with the coordinator. The coordinator, within itself, also runs a peer instance to share the download workload. In *Unity* , all the local communication among peers happens through short range radio technology such as WiFi or Bluetooth as shown in Figure 7.

4.1.1 Usage scenario: Let us now explain the utility of *Unity* through a usage scenario, as shown in Figure 7. Three friends *Alice*, *Bob* and *Carol*, are interested in the multimedia content "V" and decide to use *Unity* for collaborative downloading as follows:

1. *Alice* decides to be the initiator and therefore launches *Unity* coordinator mode. She finds the URL of "V" and feeds it into *Unity*. *Bob* and *Carol* launch the peer mode of *Unity* .
2. After *Alice* chooses a network interface available within all three of their devices e.g. Bluetooth/WiFi, *Unity* launches device discovery on the selected network interface to find nearby peers—*Bob* and *Carol*, and recruit them as peers.
3. From the given URL, *Unity* in the coordinator mode finds the size of "V" using the HTTP protocol request and equally divide the file workload among the three peers by communicating with each one of them the URL address and block information. Block information denotes the number of bytes with start and end byte to be downloaded.
4. On receipt of content download request from coordinator, peers start downloading the assigned blocks using their cellular connection.
5. On complete download of the assigned individual block, each peer sends it to the coordinator who combines all the blocks to make the desired content "V".
6. Coordinator also communicates the remaining blocks (from the received and downloaded blocks) to each peer. As a result, each participating peer gets access to the whole file after combining the received blocks with its own downloaded blocks.

4.2 *Implementation details*

Initial implementation of *Unity* was built for Android phones as they continue to grow in countries like India and at the same time, provide rich networking API support essential for the implementation. However, *Unity* can work on any Java-enabled phone too. We have kept design of *Unity* modular so that it can easily run on phones with different capabilities. Some of the different modules of *Unity* are *User interface module*, *Local networking module*, *Downloader module* and *Controller module*.[32] User interface module is responsible for showing different screens to user based on the selected mode i.e. Coordinator or Peer. Figure 9a shows the home screen of *Unity* for coordinator which accepts different parameters from the user to get started. Local networking module enables seamless data sharing and message passing between different peers and coordinator using WiFi or Bluetooth. Downloader module connects to the Internet using a cellular connection and download desired content given URL address and byte ranges as an input. Also, if download of a workload got failed in between, it restarts the download of a workload from the point it got failed. Controller module invokes other modules and also based on user choice, it selects collaboration policy i.e *Unity-Adapt* or *Unity-Default* (described in Section 4.2.2).

4.2.1 Local networking: *Unity*'s local networking module may use WiFi and Bluetooth based on its availability on participating phones. However, WiFi and Bluetooth have different networking stack in the phones and thus, require completely different and independent implementation.

Unity-WiFi: This is a variant of *Unity* which uses WiFi for local communication. WiFi (802.11) supports two different modes: Infrastructure and Adhoc. In infrastructure mode, two or more WiFi enabled devices have to use a intermediate WiFi access point (AP) to communicate between them because AP is used for routing of data packets. In adhoc mode, two different devices can directly (i.e. P2P) communicate with each other without any AP. Android started supporting WiFi Adhoc mode after OS version 4.0, popularly called as WiFi-Direct. There are large number of phones, with prior Android OS versions such as 2.2 or 2.3.[3] Building our system with WiFi-Direct would have eliminated more than 70% of the total Android based phones. Further, WiFi adhoc mode results in higher energy consumption as all the peers have to stay awake and send beacons to exchange data whenever required.

As an alternative of WiFi adhoc mode, we use a novel utility provided by Android called as WiFi hotspot, primarily designed for sharing the Internet connection of the phone with other devices such as a laptop. WiFi hotspot utility is available on all version of Android which are running Android 2.3 or beyond. WiFi hotspot utility uses 802.11 infrastructure mode which turn the phone into WiFi AP and other phones[b] can connect to it. For simplicity, let us assume that coordinator is acting as WiFi AP and all other phones connected to it are different peers. Figure 8 shows various control and data exchanges between a *Unity* coordinator and two *Unity* peers, following is corresponding description:

1. The phone, which is running *Unity* coordinator creates WiFi AP and other peers connect to it as clients.
2. As shown in Figure 8, coordinator launches device discovery to discover all connected peers and exchange a few control messages with them individually to get information such as peer name. (Refer Phase 1)
3. After device discovery step, block information and URL is passed on to all the peers using a control message and each of them start downloading their block from Internet. By default, Android uses WiFi AP functionality for enabling tethering and it may happen that peers start downloading using coordinator's data connection. To force the peers to use their own data connection, we change the connection priority during download. (Refer Phase 2)

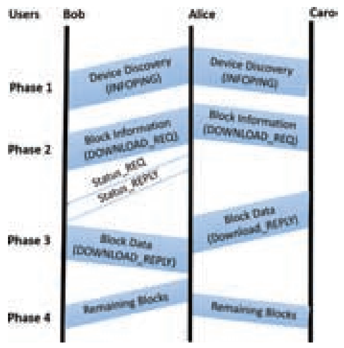
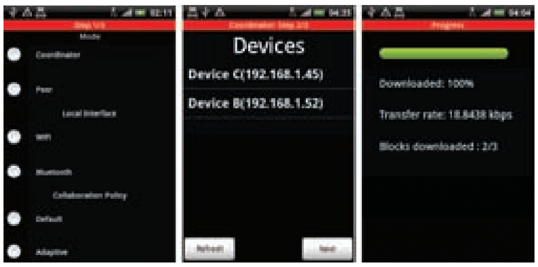

Users
Bob
Alice
Carol
Phase 1
Phase 2
Phase 3
Phase 4
Device Discovery (INFOPING)
Device Discovery (INFOPING)
Block Information (DOWNLOAD_REQ)
Block Information (DOWNLOAD_REQ)
Status_REQ
Status_REPLY
Block Data (DOWNLOAD_REPLY)
Block Data (Download_REPLY)
Remaining Blocks
Remaining Blocks

Figure 8: Sequence diagram of various control and data exchanges between different phones in *Unity-WiFi*.



(a) Home Screen (b) Device Discovery (c) Progress Status

Figure 9: Different screens for *Unity* coordinator: (a) shows the different modes and variants of *Unity*, (b) Peers running *Unity* peer mode and (c) Transfer rate of downloading and blocks received from other devices.

[b] Android 2.3 based AP can support up to 6 connected devices whereas Android 4.0supports up to 7 devices.

4. Coordinator can also check the status of block download in between by sending a status request.
5. On download completion, peers send their data blocks to the coordinator. On receipt of blocks from all the peers, coordinator sends the remaining blocks for each peer to them. Peers then merge the received blocks with downloaded blocks to get the complete content. (Refer Phase 3 and 4)

Coordinator, acting as WiFi AP, will be awake for whole download duration while the peers can operate in power saving mode (PSM) which consumes negligible energy[29] or even turn off their WiFi to save energy when not in use. As shown in Figure 7, star topology where one phone, acting as coordinator, communicates with all the other phones results in smaller local communication bandwidth as compared to the distributed architecture. *Unity-WiFi* requires that at least one person in the group should have a phone with WiFi hotspot capability and all other phones should have WiFi.

Unity-Bluetooth: To enable *Unity* on feature phones, we also developed a Bluetooth based local networking module. Bluetooth only supports adhoc P2P connection. In case of *Unity*, coordinator runs Bluetooth server instance and the peers run Bluetooth client instance. All control and data exchanges in *Unity-Bluetooth* happen in the same order as *Unity-WiFi*. In the device discovery phase, each *Unity* peer creates a bluetooth socket with a service record[c] and listens for incoming connections whereas *Unity* coordinator connects with them subsequently and exchanges information such as peer name as shown in Figure 8. Bluetooth server stores all the UUIDs with peer names for futurecommunication. Unlike *Unity-WiFi*, it does not require changing data priority on different peers. nication.

4.2.2 Collaboration schemes: *Unity* has two different collaboration schemes—*Unity-Default*, *Unity-Adapt*. *Unity-Default* divides the desired workload of size *d* into equally sized blocks. If there are *n* devices participating in the download, block size, to be downloaded by eachpeer, will be *d*/*n*. This scheme has advantage in terms of fairness as all the collaborating peers will incur equal amount of data connection expense. However, in cellular network conditions, it is usual that some nearby peer may be experiencing poor cellular network conditions resulting in low download rate.[39] In such cases, this scheme will result in increased waiting time for *Unity* coordinator and other peers due to the peer who is experiencing low throughput. Our experiments showed that for large downloads, this incremental wait could be several minutes thus correspondingly increasing the energy consumption as well.

To reduce this waiting time, *Unity* uses a simple algorithm, which adapts to changing network conditions termed as *Unity-Adapt*. For a workload of size *d*, *Unity-Adapt* divides it into equally sized blocks of size *k*.[d] *Unity* coordinator assigns each peer a single block to download at a time and the peer is expected to request another block to download whenever it finishes downloading 80% of the assigned block. *Unity* coordinator will keep on allocating the blocks dynamically until all the blocks are assigned. Thereafter, *Unity* peers will send all the downloaded blocks to *Unity* coordinator together to minimize control overhead and frequent connections.

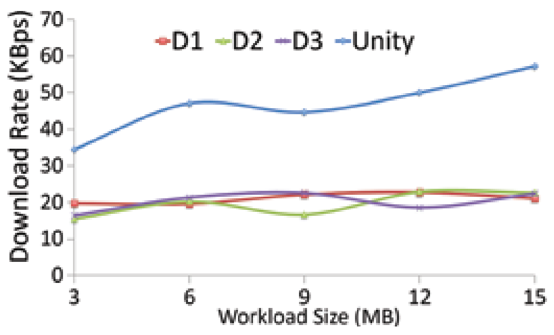
4.3 *Evaluation*

In this section, we present brief evaluation results of *Unity* while running on Android phones. The detailed evaluation results are given in our prior work.[32] We define some of the evaluation metrics for *Unity*. *Total download time* is the time taken by *Unity* to collaboratively download a workload and deliver it to all the collaborating peers. From *total download time*, we compute *effective download rate* which is equal to workload size divided by *total download time*. Our evaluation experiment consists of four Android phones, three of them manufactured by HTC and one by Samsung. All the phones were running Android 2.3.3 OS.

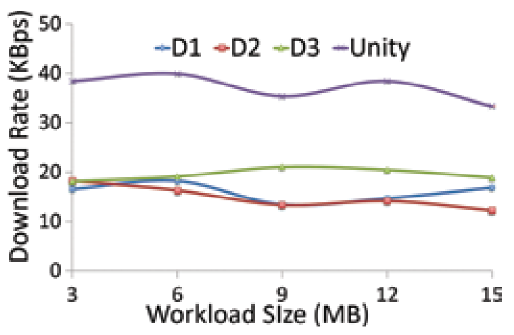
4.3.1 Download rate vs workload size: To evaluate download rate in *Unity* with varying number of collaborating devices and varying workload sizes, we downloaded five different workloads i.e. 3 MB, 6 MB, 9 MB, 12 MB, 15 MB with default collaboration policy. Number of collaborating devices were varied from 2, 3 and 4 for each of the workload. For each download instance, the download rate of individual devices are computed from the time taken by them to download the assigned workload and effective download rate of *Unity* is computed as defined in metrics above. In case of *Unity-WiFi* with 3 devices, as shown in Figure 10a, *effective download rate* increases

[c] *Unity* has a common service name and unique UUID number for each peer.

[d] Value of k in this case is typically greater than the number of collaborating devices.



(a) Download rate with 3 devices using *Unity-WiFi*



(b) Download rate with 3 devices using *Unity-Bluetooth*

Figure 10: Download rate of *Unity-WiFi* and *Unity-Bluetooth* with different workloads and total 3 collaborating devices. *D*1, *D*2, and *D*3 represents the individual device's estimated download rate.

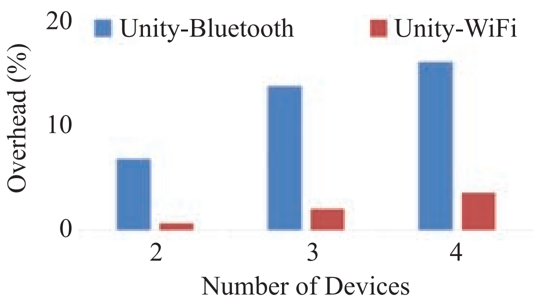


Figure 11: Overhead % comparison between *Unity-WiFi* and *Unity-Bluetooth*.

linearly with workload size. *Unity* download rate is comparatively low for smaller workloads as local communication overhead for collaboration across different peers takes significant time. However, with increasing workload size, this overhead becomes negligible.

4.3.2 Overhead comparison: *Total download time* for *Unity* consists of workload downloading time from internet, local sharing amongst collaborating devices and merging the shared workloads. It is useful to accurately quantify the overhead caused by different *Unity* operationsi.e. local networking and merging, and compare them with the *total download time*. For this purpose, we ran three instances of workload (12 MB) using *Unity* and collected logs with high resolution time intervals for these activities. Average overhead % across the 3 instances for *Unity-WiFi* and *Unity-Bluetooth* is shown in Figure 11.

We observed that much of the overhead in *Unity* is dominated by local networking module for exchanging control and data messages across different devices. As a result, overhead % using WiFi is smaller than using Bluetooh due to the corresponding difference in data transfer rates (WiFi: 1.5–2 MBps and Bluetooth: 450–480 KBps).[29]

4.3.3 Measuring impact of *Unity-Adapt*: Due to variable cellular network conditions, one or more devices may download at a lower rate in *Unity* thereby increasing the overall download time. As an instance, in Figure 10b, device *D*3 downloaded with slower rate as compared to the other 2 devices. To avoid such a situation, *Unity-Adapt* divides the whole workload into smaller block sizes and keeps assigning them to the collaborating peers based on their download rate. Empirically, we found that block size equal to 1 MB works well in *Unity* and used it for these experiments. With 3 collaborating devices, we gave *Unity* a workload of 12 MB to download in three different instances. Across all of these instances, average download rates of the three devices were 5.94 KBps, 8.14 KBps and 10.54 KBps for *D*1, *D*2 and *D*3 respectively. On an average, *Unity* without adaptation downloaded the whole workload in approx. 692 seconds. However, when using *Unity-Adapt*, *total download time* was reduced to approx. 505.78 seconds resulting in approx. 27% improvement. Additionally, the workload downloaded by a peer on an average was representative of their download rate i.e. D1 (3 MB), D2 (4 MB) and D3 (5 MB).

5 Sensing

Dedicated sensing infrastructure does not exist in many countries due to lack of resources, cost, and bigger scale of deployment. Smartphones have many sensors such as accelerometer, audio, GPS, camera etc which can be used for collecting rich and good quality data with minimal cost as compared to dedicated sensors deployment. As discussed earlier, penetration of smartphones is limited for now. Hence, our participatory sensing system design have multimodal interfaces to submit important events related to city as shown in Figure 12. Apart from "human-in-loop" data collection interfaces, our system automatically extracts events from given social media feeds. All the submitted events go to the Cloud which aggregates, pre-processes these events, and then, find patterns from the data. Moreover, our system is designed for unified open-ended sensing and broadly divide all events into five major categories: Civic complaints, traffic, neighbourhood issues, emergency and others.

5.1 *System details and deployment*

In *Human Sensors* system, there are different modes to submit event reports i.e. (1) mobile application, (2) SMS, (3) Web-based form and it automatically extracts event reports from social media feeds.[45]

5.1.1 Android-based mobile application: We built an application for only Android OS due to two different reasons; (1) It provides rich support of APIs to capture contextual data, (2) Android based mobile devices are getting increasingly popular in developing countries such as India. The android application is designed in such a way that it provides a user friendly UI for the participants to report events with minimum efforts. Snapshot of different screens of the application can be seen on Google play.[11]

Whenever, a user wants to submit an event, she chooses a category which broadly describe the event among civic complaints, traffic, neighbourhood issues, emergency or others. After choosing an appropriate category of the event, application prompts user to enter more details about the event i.e. free form text describing the event, the location/landmark of the event, and some appropriate tags related to the event. To provide more contextual sensor information, which can further assist the event report, the participant can also click the button *Click Image* which starts the camera of the phone and captures an image. The *Record Audio* button records a short audio clipping of 10 second duration to capture the sound in the vicinity of the event. On pressing the *Submit* button all the data including the text input, image, audio clip along with the GPS coordinates and cell information is uploaded using HTTP Post request to a server. Based on HTTP response, users get a notification on their phone either acknowledging successful upload or an error message incase of a failure.

5.1.2 SMS and web based event report submission: There are significant number of phones which do not run Android OS or limited programming capability. To extend the reach of our event report submission, we enabled participant to send report via SMS messages too. This option is suitable for non-programmable phones, non-supported smartphones, and users who do not prefer to use their data connection for sending reports.The following is a sample report: *Police*

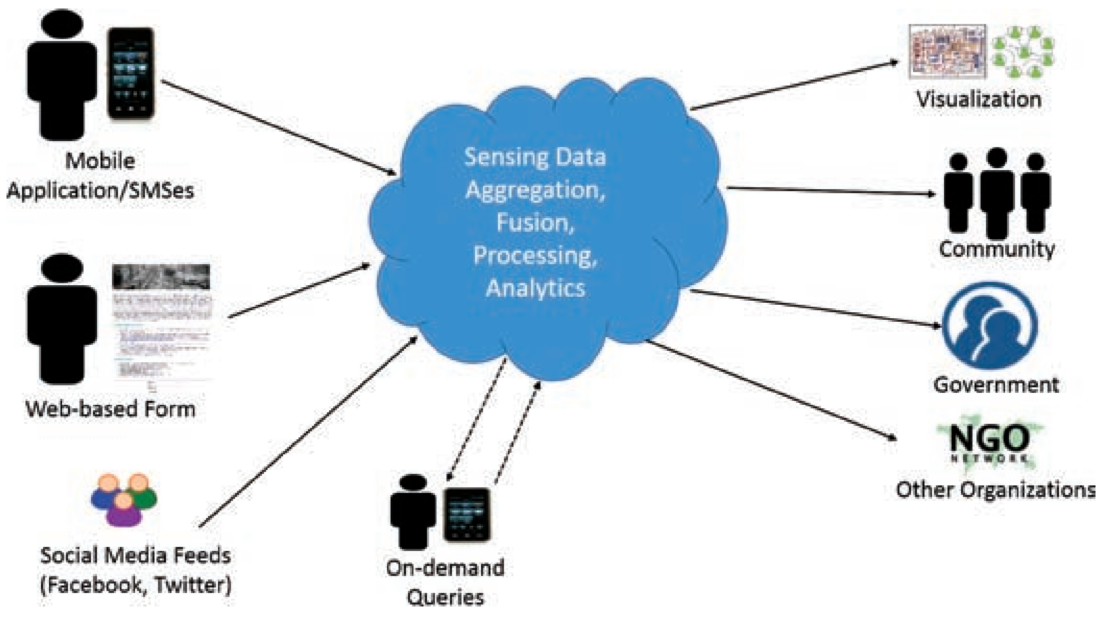


Figure 12: CrowdSensing and Multi-modal Data Fusion Testbed.

asking 1000 rs bribe for approving passport for a friend, though all documents are perfect @ indore

For non-Android smartphone or tablet users, we have also enabled web based event report submission. After one time registration and login, user can go to the website[12] and submit the event report similar to Android application. Our web-based form too allows rich data collection which can consist of text, audio and video inputs.

5.1.3 Social media feeds: Specifically in India, there are some initiatives started by government departments[6] and individuals[18] to crowdsense and disseminate information which may benefit citizens. Our system uses APIs provided by social mediawebsites such as Facebook to extract event reports from them which may complement the data reported by testbed users.

5.2 *System deployment*

We have publicized our system among university students through email and posters in India and asked them to submit city events happening around them. There were also facebook and twitter pages to continuously engage the students. The duration of system deployment was for about 3 months. Due to multiple event report submission methods, we capture different kind of information among which some of them need to be input by the user, while others get automatically captured and are sent when the event is submitted as shown in Table 6.

From our deployment, we were able to collect a total of 838 event reports among which 488 were submitted using web, 210 were submitted using SMSes and 140 events were submitted using Android application. A total of 435 users were registered in the system. While participants can submit text, audio and images, predominantly they have submitted text details of the events. Overall, we got 838 text reports i.e. all the submitted events had text, 182 of events had images and 5 had audio clip.

Table 6: Automatic and manual information collected by different modes of our crowdsensing system i.e. SMS(S), Web-based form (W), Mobile Application (MA) and Social Media Feeds (SM)

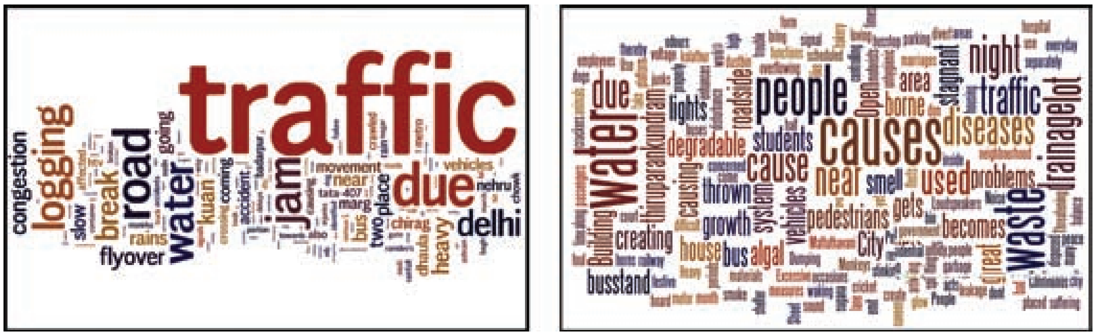
Automatically sent details	Manually sent details
Time Stamp (S + W + MA + SM)	Event Type—(W + MA)
Lat, Lng, Cell Info (MA + SM)	Message/Text (S + W + MA + SM) Event Tags (W + MA) Textual Location (S + W + MA + SM) Image & Audio (W + MA + SM)

5.3 *Preliminary data analysis*

Most of text reports submitted using mobile application or SMSes contained noisy data because intentional corruptions are very common in data uploaded from mobile devices.[42] This is due to the limited data entry options (keypad constraints on mobile devices) and due to the pressure of reducing communication latency (or cost in case of SMS) by keeping messages short yet intelligible. We have used several pre-processing steps to extract meaningful information such as location, category etc from the reports which came from social media or SMS. For instance, many text reports do not have any delimiter which can be used to find the location names embedded into it. To find location name in such reports, we parse the text and usepopular location suffixes such as 'nagar', 'chowk', 'cross' etc to estimate the location name. We have used a location dictionary to automatically learn location suffixes which can help in location name extraction. Extracted location name could be a locality name or a city name, we used Google's geocoding API to convert the location names to approximate geo-coordinates. But, there may exist some location names which can not be geocoded by Google's geocoding service i.e. some lesser known locality names which does not exists on Google Maps.[47] For such names, we used a dictionary of 963 Indian cities and towns to translate the location names to their corresponding city names. Those city names were in turn fed to the Google's geocoding service to retrieve approximate geo-coordinates.

As part of analysis, we analyzed the distribution of different event categories across different states for finding major patterns in the submitted data. Across India, most of events (43%) submitted are about civil issues followed by traffic issues (22%), neighborhood issues (18%), emergency (7%) and others (10%). The three states from which we have received the maximum data are Uttrakhand (105), Delhi (95) and Tamilnadu (80). We have seen a large variance in distribution of various event categories across different states, for instance, maximum reports (54%) from Uttrakhand contains civil complaints; in Delhi, traffic events (42%) and from Tamilnadu, both civic and neighborhood issues (80%) were maximum.

Further, we analyzed the event text for specific categories to find broad patterns in the data. Figure 13a shows the tag cloud of textual reports submitted for traffic related events in Delhi. Most of the event deals in reporting of high



(a) Traffic Issues in Delhi (b) Neighborhood Issues in Tamilnadu

Figure 13: Tag Clouds of some of major patterns found in data collection, Delhi residents were mostly concerned with traffic-related problems where as Tamilnadu had civic issues.

traffic, jam, congestion, water logging etc. Similarly, Figure 13b shows the tag cloud of report submitted for neighborhood issues in Tamilnadu and most of them deals with water logging, waste, drainage, unauthorized parking etc. From our preliminary data analysis, we have found that *Human Sensors* can potentially be used to source data from community and then making decisions accordingly. From our limited data, we found that Delhi is more concerned about traffic jams, a more detailed analysis of data can also reflect the causes of traffic jams as well as location where they happen regularly.

6 Discussion

Sixty percent of total phones will be feature phones in 2016. Due to absence of many sensors and good processing capacity, many feature phone users can not use applications which have become ubiquitous among smart phone users. In this paper, we take into consideration three different directions i.e. Localization, Communication, and Sensing to empower feature phones. For each of these directions, we describe various challenges in realizing a working system and design a system solving some of those challenges.

For Localization, we described our CBS-based approach which removes the necessity of war-driving or building a Cell ID database for GSM based localization. Evaluation using real-world traces shows that the proposed approach can provide reasonably good accuracy which is sufficient for many location based services. We have developed several location-aware application using CBS-based localization technique and even built multimodal techniques using Cell ID and GPS, which can minimize energy consumption on smartphones. Hence, CBS-based localization is a promising solution, especially for feature phones and provides mobile users in developing countries, an opportunity to access location based services without any extra infrastructure.

For Communication, we described *Unity* which enables faster communication on several co-located phones which have limited bandwidth connection (2G). Multiple people, specifically those who have similar interests typically inferred by social network or geographic proximity, have overlapping interests in desired content such as multimedia songs and videos. However, most often they tend to download the same content individually from Internet. *Unity* enables collaboration between co-located and socially connected users to download mutually desired content from Internet. *Unity* is implemented as a complete system for Android and is evaluated for effectiveness on different workload sizes and varying number of collaborating devices. *Unity* users will benefit byincurring lower costs for data connection as well as multi-fold increase in download time while reducing overall energy consumption. However, users have to manually keep track of their friend's content preference as well as location in *Unity* . We are extending current architecture of *Unity* with the help of the cloud to address some of these limitations and make collaboration more useful and pervasive. The cloud acts as a control information gateway among different mobile peers interested in collaboration.

For Sensing, we designed and deployed an open-ended community sensing test bed in India. The main goal of test bed was to sense various events across different cities or day-to-day problems with citizens' participation. We have enabled multimodal submission interface for submission of events to increase citizen participation. Submission interfaces such as SMS gives flexibility as well as cost reduction to participants because many people in developing countries subscribe to bulk SMS plans. From our deployment, we have observed that all our submission interfaces were used by the participants. The primary goal of our effort was to explore the challenges in collecting balanced and reliable data by exploiting the unreliable, autonomous "community of sensors". This was a preliminary effort and we look to solve many challenges which we encountered during our deployment such as engaging participants, controlling data quality and automatic data validation.

Other the afore-mentioned three aspects, one more aspect, where there is a disparity between feature phones and smartphones is computational resources. One way to deal with the disparity is with the use of cloud-based VM. In this solution, the feature phone is made into a thin client and the compute intensive operations are done in a VM on the cloud. We are currently working on this solution.

Acknowledgements

First author is supported by a PhD Fellowship from Microsoft Research, India. The work presented in this paper is outcome of many collaborations and research grants including Nokia Research Centre, Microsoft Research India, and IBM India Research Labs. We would like to thank all the volunteers, who helped us in collecting data for all these projects and Aparna Bharati for proof reading drafts of this paper.

Received 9 July 2013.

References

1. 3G penetration statistics in China, 2013. http://in.news.yahoo.com/billion-mobile-phones-china-3g-penetration-low-042833846.html.
2. 3G penetration statistics in India, 2013. http://www.zdnet.com/in/3g-subscribers-form-2-percent-of-indias-mobile-users-7000004883/.
3. Android OS Version Statistics, 2013. http://androidandme.com/2012/05/smartphones-2/ android-4-now-on-5-of-android-devices-gingerbread-still-dominant/.
4. Cell Broadcast Standards, 2013. http://cell-broadcast.blogspot.com/2005/11/history-and-importance-of-cell.html.
5. Cell Spotting, 2013. http://www.cellspotting.com/webpages/ cellspotting.html.
6. Delhi Police Facebook Traffic Page, 2013. https://www.facebook.com/pages/Delhi-Traffic-Police/117817371573308fref=ts.
7. Google Geocoding APIs, 2013. https://developers.google.com/maps/documentation/geocoding/.
8. Google Mobile Maps, 2013. http://maps.google.com.
9. Google Place APIs, 2013. https://developers.google.com/places/ documentation/.
10. Google Place APIs, 2013. https://maps.googleapis. com/maps/api/place/search/json?location=28.5962491,77.3396212&radius=5000&name=TRILOKPURI&sensor=false&key=AIzaSyB5lTaawAGOMSLTyiQGJBwlWt4b4C-4iUc.
11. Human Sensors Android Application on Google Play, 2013. https://play.google.com/store/apps/details?id=com.ibm.sensingApplication.
12. Human Sensors Web Application, 2013. http://kalpa.haifa.il.ibm.com: 9080/indiaChallenge/.
13. Introduction to Cell Broadcast, 2013. http://www.gsm-helpdesk.nl/en/ helpdesk/helpdesk.php?id=57.
14. Mobile only users across the world, 2013. http://mobithinking.com/mobile-marketing-tools/latest-mobile-stats/b#mobile-only.
15. Number of mobile phone shipments, 2013. http://mobithinking.com/mobile-marketing-tools/latest-mobile-stats/a#phone-shipments.
16. Number of mobile phone subscriptions , 2013. http://www. internetworldstats.com/mobile.htm.
17. Open Cell ID, 2013. www.opencellid.org.
18. Power Cuts, 2013. www.powercuts.in.
19. P. Bahl and V. N. Padmanabhan. Radar: an in-building rf-based user location and tracking system. pages 775–784, 2000.
20. A. Balasubramanian, R. Mahajan, and A. Venkataramani. Augmenting mobile 3g using wifi. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pages 209–222. ACM, 2010.
21. N. Balasubramanian, A. Balasubramanian, and A. Venkataramani. Energy consumption in mobile phones: a measurement study and implications for network applications. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, pages 280–293. ACM, 2009.
22. M. E. N. Bayir, M. A.; Demirbas. Discovering spatiotemporal mobility profiles of cellphone users. In *IEEE International Symposium on World of Wireless, Mobile and Multimedia Networks and Workshops, 2009. WoWMoM 2009*, Washington, DC, USA, 2009. IEEE Computer Society.
23. A. T. Campbell, S. B. Eisenman, N. D. Lane, E. Miluzzo, R. A. Peterson, H. Lu, X. Zheng, M. Musolesi, K. Fodor, and G.-S. Ahn. The rise of people-centric sensing. *Internet Computing, IEEE*, 12(4):12–21, 2008.
24. M. Y. Chen, T. Sohn, D. Chmelev, D. Haehnel, J. Hightower, J. Hughes, A. Lamarca, F. Potter, I. Smith, and A. Varshavsky. Practical metropolitan-scale positioning for gsm phones. In *In Proceedings of the Eighth International Conference on Ubiquitous Computing*. Springer, 2006.
25. M. Y. Chen, T. Sohn, D. Chmelev, D. Haehnel, J. Hightower, J. Hughes, A. LaMarca, F. Potter, I. Smith, and A. Varshavsky. Practical metropolitan-scale positioning for gsm phones. In *UbiComp 2006: Ubiquitous Computing*, pages 225–242. Springer, 2006.

26. Y.-C. Cheng, Y. Chawathe, A. LaMarca, and J. Krumm. Accuracy characterization for metropolitan-scale wi-fi localization. In *Proceedings of the 3rd international conference on Mobile systems, applications, and services*, pages 233–245. ACM, 2005.
27. J. Eriksson, L. Girod, B. Hull, R. Newton, S. Madden, and H. Balakrishnan. The pothole patrol: using a mobile sensor network for road surface monitoring. In *ACM MobiSys*, 2008.
28. S. Feng and C. L. Law. Assisted gps and its impact on navigation in intelligent transportation systems. In *Intelligent Transportation Systems, 2002. Proceedings. The IEEE 5th International Conference on*, pages 926–931, 2002.
29. R. Friedman, A. Kogan, and Y. Krivolapov. On power and throughput tradeoffs of wifi and bluetooth in smartphones. In *INFOCOM, 2011 Proceedings IEEE*, pages 900–908. IEEE, 2011.
30. S. Gaonkar, J. Li, R. R. Choudhury, L. Cox, and A. Schmidt. Micro blog: sharing and querying content through mobile phones and social participation. In *Proceeding of the 6th international conference on Mobile systems, applications, and services*, MobiSys '08, pages 174–186, New York, NY, USA, 2008. ACM.
31. M. Ibrahim and M. Youssef. A hidden markov model for localization using low-end gsm cell phones. In *Communications (ICC), 2011 IEEE International Conference on*, pages 1–5. IEEE, 2011.
32. P. Jassal, K. Yadav, A. Kumar, V. Naik, V. Narwal, and A. Singh Unity: Collaborative downloading content using co-located socially connected peers. In *The Ninth International Workshop Mobile Peer-to-Peer Computing (MP2P'13)*. IEEE, 2013.
33. M. Kjasrgaard. Location-based services on mobile phones: Minimizing power consumption. *Pervasive Computing, IEEE*, 11(1):67–73, 2012.
34. N. D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. T Campbell. A survey of mobile phone sensing. *Communications Magazine, IEEE*, 48(9):140–150, 2010.
35. P. Mohan, V. N. Padmanabhan, and R. Ramjee. Nericell: rich monitoring of road and traffic conditions using mobile smartphones. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 323–336. ACM, 2008.
36. M. Mun, S. Reddy, K. Shilton, N. Yau, J. Burke, D. Estrin, M. Hansen, E. Howard, R. West, and P. Boda. Peir, the personal environmental impact report, as a platform for participatory sensing systems research. In *Proceedings of the 7th international conference on Mobile systems, applications, and services*, pages 55–68. ACM, 2009.
37. P. Nurmi, S. Bhattacharya, and J. Kukkonen. A grid-based algorithm for on-device gsm positioning. In *Proceedings of the 12th ACM international conference on Ubiquitous computing*, pages 227–236. ACM, 2010.
38. J. Paek, J. Kim, and R. Govindan. Energy-efficient rate-adaptive gps-based positioning for smartphones. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, MobiSys '10, pages 299–314, New York, NY, USA, 2010. ACM.
39. A. Schulman, V. Navda, R. Ramjee, N. Spring, P. Deshpande, C. Grunewald, K. Jain, and V. N. Padmanabhan. Bartendr: a practical approach to energy-aware cellular data scheduling. In *Proceedings of the sixteenth annual international conference on Mobile computing and networking*, pages 85–96. ACM, 2010.
40. A. Schwaighofer, M. Grigoras, V. Tresp, and C. Hoffmann. Gpps: gaussian process positioning system for cellular networks. In *IN NIPS*. MIT Press, 2004.
41. A. Sharma, V. Navda, R. Ramjee, V. N. Padmanabhan, and E. M Belding. Cool-tether: energy efficient on-the-fly wifi hot-spots using mobile phones. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, pages 109–120. ACM, 2009.
42. L. V. Subramaniam, S. Roy, T. A. Faruquie, and S. Negi. A survey types of text noise and techniques to handle noisy text. In *Proceedings of The Third Workshop on Analytics for Noisy Unstructured Text Data*, pages 115–122. ACM, 2009.
43. A. Thiagarajan, L. S. Ravindranath, H. Balakrishnan, S. Madden, L. Girod. Accurate, Low-Energy Trajectory Mapping for Mobile Devices. In *8th USENIX Symp. on Networked Systems Design and Implementation (NSDI)*, Boston, MA, March 2011.
44. L. Vu, Q. Do, and K. Nahrstedt. Jyotish: A novel framework for con structing predictive model of people movement from joint wifi/bluetooth trace. In *Pervasive Computing and Communications (PerCom), 2011 IEEE International Conference on*, pages 54–62. IEEE, 2011.
45. K. Yadav, D. Chakraborty, S. Soubam, N. Prathapaneni, V. Nandakumar, V. Naik, N. Rajamani, L. V. Subramaniam, S. Mehta, and P. De. Human sensors: Case-study of open-ended community sensing in developing regions. In *11th International Conference on Pervasive Computing and Communications*. IEEE, 2013.
46. K. Yadav, V. Naik, and A. Singh. Mobishare: cloud-enabled opportunistic content sharing among mobile peers. 2012.
47. K. Yadav, V. Naik, A. Singh, P. Singh, and U. Chandra. Low energy and sufficiently accurate localization for non-smartphones. In *Mobile Data Management (MDM), 2012 IEEE 13th International Conference on*, pages 212–221. IEEE, 2012.
48. K. Yadav, V. Naik, A. Singh, P. Singh, P. Kumaraguru, and U. Chandra Alternative localization approach for mobile phones without gps. In *In proceedings of NSDR'10 (co-located with Mobisys'10)*, 2010.
49. K. Yadav, V. Naik, P. Singh, and A. Singh. Alternative localization approach for mobile phones without gps. In *Middleware '10 Posters and Demos Track*, Middleware Posters '10, pages 1:1–1:4, New York, NY, USA, 2010. ACM.
50. Y. Zhao. Mobile phone location determination and its impact on intelligent transportation systems. *Intelligent Transportation Systems, IEEE Transactions on*, 1(1):55–64, Mar 2000.

Kuldeep Yadav is a 4th year Ph.D. student in Mobile and Ubiquitous group at IIIT-Delhi. His current research focusses on building large-scale mobile systems for developing countries with specific focus on user location. He has been awarded prestigious Microsoft Research Ph.D. Fellowship in 2011, which is given to only 5 computer science Ph.D. students every year in India. His research is presented and demonstrated at various conferences i.e. ACM MobiSys, ACM/USENIX Middleware, IEEE MDM, IEEE PerCom, and ACM HotMobile. He has also been awarded best presentation award in Ph.D. Forum at MobiSys'12 sponsored by Google and 1st runners up award in Ph.D. poster session at Microsoft Research Tech Vista.

Vinayak Naik is an Assistant Professor at Indraprastha Institute of Information Technology, Delhi (IIIT-Delhi) since January 2010. He received his Ph.D. in CSE from Ohio State University, USA in 2006 and BTech from VJTI, India in 1999. He has worked at IISc, UCLA, Telcordia, and TCS in the past. His research focuses on mobile computing and reliable large-scale wireless/sensor networks. He was honored with Rajiv Gandhi Excellence award in Aug 2011 and CENS Local Employee Award in Aug 2008. His Ph.D. students have been awarded Microsoft Research Ph.D. Fellowship, Google Best Presenter Award at MobiSys'12 conference, and Prime Minister's Fellowship Scheme for Doctoral Research.