Crowd-sourced sensing and collaboration using twitter — Source link

Crowd-Sourced Sensing and Collaboration Using Twitter

Murat Demirbas, Murat Ali Bayir, Cuneyt Gurcan Akcora, Yavuz Selim Yilmaz
Computer Science & Engineering Dept.,
University at Buffalo, SUNY
Email: {demirbas, mbayir, cgakcora, yavuzsel}@cse.buffalo.edu

Hakan Ferhatosmanoglu
Computer Science & Engineering Dept.,
The Ohio State University
Email: hakan@cse.ohio-state.edu

***Abstract*—Despite the availability of the sensor and smartphone devices to fulfill the ubiquitous computing vision, the-state-of-the-art falls short of this vision. We argue that the reason for this gap is the lack of an infrastructure to task/utilize these devices for collaboration. We propose that Twitter can provide an "open" publish-subscribe infrastructure for sensors and smartphones, and pave the way for ubiquitous crowd-sourced sensing and collaboration applications. We design and implement a crowd-sourced sensing and collaboration system over Twitter, and showcase our system in the context of two applications: a crowd-sourced weather radar, and a participatory noise-mapping application. Our results from real-world Twitter experiments give insights into the feasibility of this approach and outlines the research challenges in sensor/smartphone integration to Twitter.**

I. Introduction

The ubiquitous systems vision [1] of embedding and weaving abundantly available tiny-computers to the fabric of our daily lives is close to fruition. With the advances in MEMS technology in the previous decade, it has become feasible to produce various types of sensors (such as magnetometers, accelerometers, passive-infrared based proximity, acoustics, light, heat) inexpensively, in very small-form factor, and in low-power usage. Furthermore, cellphone technology has seen an adoption rate faster than any other technology in human history [2]: as of 2009, the number of cellphone subscribers has exceeded 3.3 billion users. The rate of innovation in this field has been head-spinning. Nokia, Google, Microsoft, and Apple have all introduced cellphone operating systems (Symbian, Android, Windows Mobile, iPhoneOS) and provided APIs for enabling open application development on the cellphones. These modern cellphones, which are dubbed as *smartphones*, enable location-aware services as well as empowering the users to generate and access multimedia content.

Despite the availability of the devices to fulfill the ubiquitous computing vision, the-state-of-the-art falls short of this vision. We argue that the reason for this gap is the lack of an infrastructure to task/utilize these devices for collaboration and coordination. In the absence of such an infrastructure, the state-of-the-art today is for each device to connect to Internet to download/upload data and accomplish an individual task that does not require collaboration and coordination. Providing an infrastructure for publish/subscribe and tasking of these devices enables any node to search the data published by several nodes in one region to aggregate and decide on a question, as well as task several nodes in one region to acquire the needed data (if the data is not already being published to the infrastructure).

We propose that Twitter [3] can provide an "open" publish-subscribe infrastructure for sensors and smartphones and pave the way for ubiquitous crowd-sourced sensing and collaboration applications. The open publish-subscribe system of Twitter implies that different actors may integrate user data differently. Moreover, third parties can use the gathered data in unanticipated ways to offer new services with them. In addition to this open publish-subscribe infrastructure, the social networks angle of Twitter also provides a useful feature for the crowd-sourced sensing and collaboration applications. Finally, the wide popularity of Twitter and the big community behind it (more than 30 million users in US), is an important reason to target our crowd-sourcing system for Twitter: It is easier to give the community a tool than to give the tool a community.

More specifically, we provide the following contributions.

- In Section II, we provide a detailed survey of Twitter with existing application domains on news and alert systems. In Section III, we present emerging application domains for Twitter: including crowd-sourcing, participatory sensing, social collaboration, expert-finding, and market research and trend mining.
- We discuss sensor integration to Twitter in Section III-A and smartphone integration in Section III-B. We point to a potential new architectural trend in sensor integration, that of inexpensive sensors using cellular data network to reach Internet in one hop.
- In Section IV, we present our design and implementation of a crowd-sourced sensing and collaboration system over Twitter. Central to our system is a Twitter-bot (with an integrated database system) that accepts questions, crowd-sources them, and aggregates the answers to reply back to the querier. The system also includes a smartphone client for automatically pushing sensor reading information to Twitter.
- In Sections V and VI-A, we showcase and evaluate the

performance of our crowd-sourced sensing and collaboration system on two case-studies. The first one is a crowd-sourced weather radar, which help monitor fine-granularity weather conditions and act as a ground-truth. [1] Our second application is noise mapping of a region by aggregating the automatic noise-sensing updates from smartphones.

- We present an analysis of our real-world Twitter experiments to give insights for the feasibility of our approach. We find that although we do not offer the user any incentives to reply, our queries receive at least 15% reply ratios. Surprisingly, 50% of the total replies arrive within the first 10 minutes of our query, and 80% of the replies arrives within the first 2 hours, enabling low-latency operations for crowd-sourcing applications. Our experiments also found that consistently the majority of replies come from users that access Twitter from their mobile phones.

II. Twitter

A. Twitter in a Nutshell

A web 2.0 project, Facebook.com established a status update field in June 2006, but it was Twitter.com that took status sharing between people to mobile phones four months later. First named as “Status”, then as “Twttr”, Twitter has gone beyond status sharing and became a Web 2.0 microblogging site for information sharing and news reporting. Twitter started its journey in 2006, but its fame started to spread after the South by Southwest festival in 2007. In the event, company set up user accounts for the participants, and used big screens streaming tweets from them in the conference simultaneously. The effect of the conference was huge for Twitter. According to a report by HubSpot in 2008 [4], despite being functional since 2006, Twitter had its 70% of users joined in 2008, and reached around 4-5 million users, making it a top 1000 website in web traffic.

The trend of growth for Twitter has continued since then. In 2009, HubSpot [5] reported an astonishing 18.0000% growth rate of Twitter users. The report gives the total number of Twitter users in USA as 27 million. 55% of these users are male, 48% is between 18 and 34 ages. Twitter has seen a rapid growth in the western sphere, and cities like London, New York and San Francisco generate the largest traffic on the site. Top 100 cities list [6] is dominated by US, with the first non-USA city being Toronto, Canada. Tehran, Iran is 18th on the list, making it the first non-western city. Tokyo, Japan, once in top 10 in 2008, now enters the list as the 21st city.

B. Beneath the Hood

Twitter’s success can be attributed to two main factors; elegance in design, and simplicity in adding third party improvements to Twitter. Elegance is due to the character limit. Twitter names micro-blog posts from users as tweets. Each tweet has a 140 character limit which is inherited from text messaging. (The original 160 character SMS limit was reorganized into 20 character username and 140 character post fields.) A comparison between blogging and microblogging gives us a good understanding of the reason behind Twitter’s popularity. While blogging requires good writing skills and large content to fill pages, Twitter restricts posts to 140 characters, which encourages much more people to post.

Simplicity is due to an early decision by Twitter to provide a HTTP based open source API and share posts with third party applications. Twitter’s API consists of two different parts: Search API and REST API. REST API enables Twitter developers to access the core Twitter data. This data includes tweets, timelines, and user data. Search API provides the developer to query the tweets. It also provides information about the trending topics. The usage of both APIs is subject to rate limiting, however, based on the request of the third party, Twitter may add these applications to its whitelist and remove request limitations. The ease and flexible usage of Twitter API encouraged several developers to write applications. Starting with Twitterrific in January 2007, many applications have been created for Twitter [7].

C. Existing Twitter Application Domains

1) News: Twitter is becoming regarded as the fastest way to reach to breaking news. Users’ collaboration has given Twitter a clear edge over news centers and recently news centers have set up Twitter accounts and encouraged users to interact with these accounts in order to capture breaking news. CNN maintains 45 official Twitter accounts with more than 5 million followers. During the election protests in Iran, Twitter played a greater role than news centers, and attracted attention. US government reportedly warned Twitter.com to not to undergo maintenance for it would break the news stream from Iranian users [8]. Even after banning foreign journalists from covering rallies, Iran could not stop information flow and finally shut down access to Twitter. In recent Mumbai attacks in India, just minutes after the attacks, Twitter was the major source until news sites caught up with updates. As well as posts, information flow to Twitter consists of pictures, links and videos. Demonstration pictures from Iran and the first picture from US Airways plane in the Hudson River [9] increased Twitter’s popularity in the public. The Economist declared Twitter a winner in this information race [10].

2) Alert Systems: Twitter provides a system that can connect residents of a city with virtually no cost. It also increases the abilities of an alert system by inputting more user generated data. Some cities already opted for Twitter to alert their residents [11]. The Virginia Tech incident in 2007 highlighted the security issues on university campuses. To integrate e2Campus emergency notification network with popular social networks, Pacific University of Forest Grove, Oregon, implemented a Twitter based alert system for its

[1] You can visit our weather radar at rainradar on Twitter. We display the answers to our weather radar on a map at http://ubicomp.cse.buffalo.edu/rainradar. The map is configurable to show results from previous days, and also is zoomable to show fine-grain locations of the replies.

students, and the trend is likely to grow [12]. With this system, universities can send e2Campus alerts to popular networks.

III. Research Directions using Twitter

In this section we discuss sensor and smartphone integration to Twitter and identify research directions and emerging applications for these domains.

A. Integrating Sensors

With the advances in MEMS technology in the previous decade, it has become feasible to produce various types of sensors (such as magnetometers, accelerometers, passive-infrared based proximity, acoustics, light, heat) inexpensively, in very small-form factor, and in low-power usage. Moreover there has been nearly a decade of research in wireless sensor networks (WSNs) and some real-world deployments of WSNs have been successfully demonstrated [13]–[16]. As such, WSNs offer an untapped source of information about our physical world. However, WSNs have not achieved the broad impact and visibility it deserves. Not only are we far away from "a central nervous system for earth", there is no significant market penetration for WSNs yet.

Arguably the greatest barrier against wider adoption of WSNs is the difficulty in locating sensors and subscribing to them. We propose that Twitter can provide an "open" publish-subscribe infrastructure for sensors, as well as the search/discovery of sensors with certain attributes. Moreover, having access to a lot of sensors is also valuable in that it would be possible to reduce false-positives from sensors by cross-checks. Below we list some ideas we are pursuing for sensor integration to Twitter.

Sensor tweet standards. In order to search and process sensor values on Twitter, we need to agree upon a standard for publishing these sensor readings. We offer a biography format on Twitter that describes a sensor in detail in Section IV-B. The bio-code makes sensors easy to find. By just searching for the desired sensor functionalities using the Twitter API over the bios, one can reach all sensors within a locality that provides the desired functionalities.

We are currently developing a standard, *TweetML*, for tweeting sensor values. We will make use of the built-in hashtags feature in Twitter for easier accessibility and searchability of sensor value fields. As part of our current work, we are publishing data to Twitter from some existing WSNs deployments. One of these is the wine-cellar monitoring WSN deployment, and another is personnel tracking WSN deployment.

New WSN architectures. The popularity of Twitter already have resulted in the production of inexpensive specialized devices for tweeting. TwitterPeek [17] is a very good example of this trend. TwitterPeek enables the user to tweet and follow Twitter from anywhere (no WiFi necessary) using the cellular data network to connect to Twitter. One can buy TwitterPeek for $199 and get connectivity service for the lifetime of the device –without any bills ever. In comparison a barebones WSN node with only 100 feet transmission radius is rated at $129. The reason TwitterPeek is able to offer such a powerful device at such a low price is because of the benefits of mass production. TwitterPeek may signal a new direction for WSN devices. Instead of using low communication range devices that incur the challenges/complexity of maintaining a multihop network and still require a basestation to access Internet, TwitterPeek-like sensors can directly reach Internet at one hop. These devices may not only tweet their sensor readings, but can also be easily controlled over Twitter to reconfigure their sensing schedules and tune their parameters.

B. Integrating Smartphones

Smartphones provide significant advantages over traditional wireless sensor nodes. Firstly, smartphones are mobile. Wherever a smartphone user goes, smartphone can take sensor readings (with built-in sensors for acoustic, image, video, accelerometer, tilt, magnetometer, and potentially with other integrated custom sensors). The dynamic geolocation feature of smartphones enables these readings to be location and time-stamped. Thus, in contrast to WSN nodes that are tied to static locations, and do not scale for coverage of large areas, smartphones cover large areas due to their mobility. Secondly, smartphones are personal and administrated by their users. In contrast to sensor networks where energy-efficiency of utmost importance, smartphones are recharged by their users and it is not necessary to try to squeeze every bit of energy. Moreover, since smartphones are personal, they provide the potential of interacting with the phone user for tasks requiring human intelligence and intervention, such as taking a picture of a requested location, answering a question for which the user is well-equipped.

Below, we identify 3 new application domains for smartphone integration to Twitter, with increasing level of complexity.

1) Participatory Sensing: Participatory sensing is the use of volunteering smartphones to collect data from a large region. Although there has been significant work on participatory sensing [18], using Twitter opens up novel improvements on this application domain. Twitter's open publish-subscribe system enables others to use the gathered data in unanticipated ways and offer new services over them. Moreover Twitter's social network aspect enables new features to be added to participatory sensing. For example, when one of the users have performed significant amount of participatory sensing but her friend and competitor (Twitter enables using lists for followers/friends) have not done anything for that week, our system can send a reminder message for that friend.

There is already good support for enabling participatory sensing applications over Twitter. Some Twitter third party applications (including Twittervision17, Twittearth, Twitter Atlas, Twibs20) use maps to show status posts, and can be configured to show posts only from certain regions.

2) Crowd-Sourcing: Crowd-sourcing means distributing a query to several Twitter users in order to gather and aggregate the results and exploit the wisdom-of-crowds effect. Examples of crowd-sourcing may be a weather/rainradar (with better precision and ground-truth than meteorological weather radars), and polling for the best restaurant entree in town.

Crowd-sourcing depends on user participation. With Twitter's popularity, finding a user to ask a question is not a problem, and we find that users are willing to participate and answer questions. In our experiments up to 1/6th of our queries got answered, although we did not provide any incentive for answering. We think this is due to the sharing and participatory nature of Twitter culture. It is possible and easy to provide incentives for encouraging participation. Using Twitter's list functionality a group of users might be classified as experts of a topic. Each topic may have multiple user groups with different expert levels. Upon answering questions, the users can get promoted to a higher level. Visibility of these lists to the public would will be a great incentive for users to collaborate. Another way to incentivize users is to give the users that answer more questions the right to send more questions to our crowd-sourcing engine.

The social network nature of Twitter can also be exploited to provide an extra incentive for crowd-sourcing. It is also possible to provide useful feedback to crowd-source participant based on others answers. For example, the participant may get to see how her answer fares with other answers. In the "best restaurant" query, participants may get to learn which other participants also favorite their restaurant of choice.

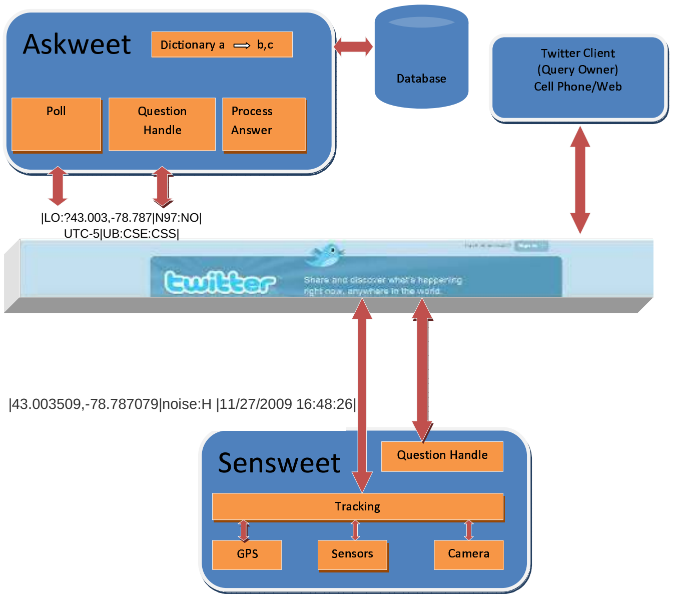
3) Social Collaboration: Social collaboration applications are more sophisticated than crowd-sourcing applications in that they require back-and-forth interaction in contrast to the asymmetric one-shot interaction involved in crowd-sourcing. Examples of social collaboration applications include pick-up soccer games, arranged ride-sharing, community-organization events, support groups for addicts, and support groups for exercising and weight-watching.

C. Data Mining of Tweets

Twitter provides an excellent medium for spatiotemporal text mining and information retrieval. Here we summarize three research problems in the context of mining Twitter data: text classification, expert finding, and trend mining.

Text Classification. A useful research problem mining of tweets is to classify streaming tweets into topic-based groups. Mining short segments of text has been studied in the literature in various other contexts, e.g., query-query similarity [19], paragraph and sentence similarity [20]. A successful Twitter text classification needs to handle a diverse set of streaming short text messages with abbreviations, slangs, and no sound grammar use. Fortunately, the quality of mining results can be improved by incorporating the rich contextual information, such as the author bio, profile, hash tags, urls, previous tweets and status of the author in the underlying social network.

Expert Finding. Expert finding have been traditionally studied in the context of enterprise intranets [21]. One of the most promising fields of information finding on Twitter takes advantage of the sheer size of its huge user base. Identifying experts in topics of user interests is a challenging task, given the large number of users and wide variety of potential interests. Some applications use bios to group similar


Askweet
Dictionary a ⟹ b,c
Poll
Question Handle
Process Answer
Database
Twitter Client (Query Owner) Cell Phone/Web
|LO:?43.003,-78.787|N97:NO| UTC-5|UB:CSE:CSS|
|43.003509,-78.787079|noise:H |11/27/2009 16:48:26|
Sensweet
Question Handle
Tracking
GPS
Sensors
Camera
Fig. 1. Crowd-sourcing System Architecture

people, and user posts can be scanned to find people with same hobbies, background and profession. Besides user bios and previous tweets as the text-base, the spatial and temporal meta-data provide a constraint on the potential user-base, since we typically look for ideas constrained to location and time. Twitter has the potential of involving more than locating experts, it provides an environment for people to assert their expertise by actively joining the information flow and giving useful insights.

Trend Analysis. While expert finding focuses on authoritative sources, observing the patterns in a crowd would provide information with the power of collaboration potentially by millions of users. Applications of trend mining include identifying and monitoring emerging topics and events dynamically [22], [23], and sentiment analysis on user posts for products publicized on Twitter [24]. Canonizing some ideas through Twitter user posts has an inherent liability to manipulation, but it also offers a quick and effective way of getting to know how people react to, discuss and adopt new ideas. By aggregating users' ideas, we can effectively eliminate fringe cases, and find accurate information on a fact. The system strongly resembles the idea of democracy. Crowd mining is a luminous manifestation of the power of Web 2.0 applications. To make it more interesting, Twitter as an open platform enables briefer exchanges of information that would be lost in a lengthy blog or text.

IV. Our Crowd-sourcing System Architecture

In this section, we present the design of our crowd-sourced sensing and collaboration system over Twitter. Figure 1 illustrates the high level architecture of our crowd-sourcing system. Twitter acts a middleware for publish/subsribe as well as search & discovery. Our system is composed of three components namely *Askweet*, *Sensweet* and *Twitter clients*. *Sensweet* is a smartphone application that publishes real-time readings from the integrated-sensors to Twitter. *Askweet* is a program that listens to its Twitter account for questions and processes the questions and aggregates the replies it receives to these questions from *Sensweet* and the *Twitter clients*. We discuss the design of the Askweet and Sensweet components in more detail below.

A. Askweet

Askweet accepts a question, and tries to answer the question using the data on Twitter, potentially data published by Sensweets. If it is not possible to answer the question with existing data and/or if the question requires interaction, Askweet finds experts on Twitter (potentially using information retrieval techniques) and forwards the question to these experts. After obtaining answers from the experts, it replies the answers back to the asker. Askweet accepts a certain syntax from queries and replies, but it can also be extended and generalized to adopt modern natural language processing techniques.

The Askweet components of two case studies in this paper run on a dedicated server, and keep all relevant data in a database to process questions and replies in a coordinated matter. Due to the parallelizable nature of processing queries and replies (a thread is assigned to each reply), it is easy to deploy Askweet on a cloud computing platform for elastic scalability. Since Askweet accounts have been recently whitelisted by Twitter and hourly request limits removed, it is possible to implement Askweet over Hadoop Map/Reduce framework to handle millions of queries and replies daily.

B. Sensweet

A Sensweet application uses the smartphones' ability to work in the background without distracting the mobile user. Sensweet applications sense the surrounding environment and send these data to the Twitter. While sending the data to Twitter, the Sensweet client formats the data according to the *bio-code* it advertises in the Biography section of its Twitter account. The main idea of using a bio-code is to allow worldwide users to search for the sensors they are looking for on-the-fly and enjoy a plug-and-play sensor network without registering through dedicated sites.

Here we provide a standard for a bio-code for Twitter to encode the values published by the sensor. To illustrate with an example, the Bio section of our noise-sensing application reads as: $|LO:?43.003, -78.787|N97:NO|UTC-5|UB:CSE:CSS|$. This bio-code consists of tuples separated with a vertical bar ($|$). In each tuple, descriptive fields are separated with a colon (:). The values that are separated with commas describe the phenomena the sensor(s) captures. The first tuple is always the location parameter: longitude and latitude (obtained from the built-in GPSs or entered manually). If the sensor is mobile (e.g., smartphone), a question mark will precede the longitude value. Even for mobile sensors a default location is added to give the queriers an idea of the region the sensor operates. The question mark hints that a more accurate location is included in the tweets. The second tuple explains the manufacturer of the sensor, product ID (if possible) and the sensor type(s) the

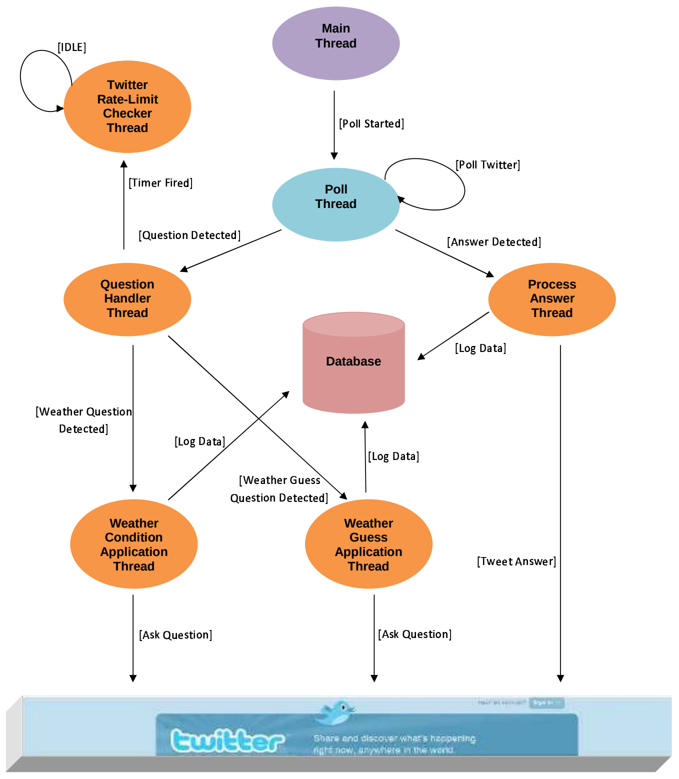


Fig. 2. State transition diagram for Askweet component

sensor provides. The third tuple is optional, and describes the time zone that the sensor uses and can also include a timestamp. Although Twitter provides timestamping of tweets, this extra timestamp becomes important in case when a sensor need to store readings and send them later when it can connect to the Internet. The fourth tuple involves identification of the company/project that deploys the sensor, and defines a group id to locate other sensors that are part of that project.

Thus, the above bio-code is decoded as: Location is dynamic, but default location is UB North Campus Bell hall, Nokia N97 is used to capture GPS and accelerometer values in NY time zone for UB CSE Crowd-Sourced Sensing (CSS) Project.

V. Case Study: Crowd-Sourced Weather Radar

In this section we explain our crowd-sourced weather radar application. For the sake of simplicity, we choose a topic where everybody in Twitter can be an expert: the current weather condition. Our application contains two sub-applications, one of them obtains the current weather condition from users, and the other one obtains guesses from the users about the next day's weather condition.

Weather radar application has its own question and answer format. The question messages sent by query owners are in the form of "?[Application Name] Loc:Location" where application name is either Weather or WeatherGuess. For instance "?Weather Loc:Buffalo,NY" might be a valid question for asking weather condition in Buffalo,NY. The forwarded query to the Twitter users is of the form: "How is the weather there now? reply 0 for sunny, 1 for cloudy, 2 for rainy, and 3 for snowy" Our weather radar application account can be visited at rainradar on Twitter. We display the answers to our weather radar on a map at http://ubicomp.cse.buffalo.edu/rainradar. The map is configurable to show results from previous days, and is also zoomable to show fine-grain locations of the replies.

We have implemented only the Askweet component of the crowd-sourced system since the Sensweet component can be any Twitter client. The Askweet component of our weather radar application is written in Java Programming language by using Twitter4J open source API library and total size of the source is about 2KLOC. Askweet listens to the incoming messages to its Twitter account and processes them with respect to their message types. The main function of Askweet component is to get a question, process it and/or forward this query to the multiple users who can answer it. After obtaining answers from Twitter users, Askweet sends the reply to the original querier.

Our Askweet implementation is multithreaded for scalability, with each thread implementing a specific functionality. When the Askweet application is launched (Figure 2), it starts the poll thread that polls the Twitter account and gets the messages. Then the thread detects whether the message is a question or answer. Depending on the message type, it starts either a question handle thread or a process answer thread. Poll thread keeps on checking the account every minute continuously to get the new messages addressed to itself.

Question handle thread receives the question from the poll thread and detects if it is weather guess question or weather condition question. Depending on the question type it starts either a weather condition application thread or a weather guess application thread. Question handle thread also starts Twitter rate-limit checker thread in order to ensure that Askweet stays within Twitter's request limits. After this step, the question handle thread is terminated.

Weather guess application and weather condition application threads have almost the same functionality. Both of them get the question and parse the location from question text and search through Twitter to find users for the specified location. Then they send the question to the selected qualifying Twitter users. After that these application threads are terminated. Both of the applications keep all the relevant data in a database in order to observe the social collaboration and attendance. This database also helps the program not to spam any Twitter user with multiple requests within a week.

Twitter rate-limit checker thread checks the rate limit and locks question asking permit if rate limit exceeds and releases the lock if otherwise. Process answer thread gets the answers from the poll thread and tweets the answer to Twitter. It also selects five of the answers to forward to the original querier.

A. Experiment Results

In this section, we present our experimental results for weather radar application. We performed three types of experiments using weather radar. In the first one, we compare the user responses in different time slices of day for New York City (NYC). In the second, we compare user responses from three different cities: NYC, Toronto and Montreal. In the last one, we analyze the correlation of answers from our users with data from weather.com for one day (December 6, 2009).

In the first experiment, we compare the user response behaviors in NYC at different time slices. We observed that the response times in the afternoon and in the evening are better than those in the morning and at night (Figure 3(a)). An interesting phenomenon is that on the average 50% of the answers are received within the first ten minutes (Figure 3(a)). Figure 3(b) shows the user contribution to our experiments. We observe that Twitter user contribution to the experiment is highest in the morning which is nearly 20% (Figure 3(b)); we get a response from 20% of the queried users. For the other time slices, the contribution is around 15% (Figure 3(b)). Figure 3(c) shows the user distribution with respect to Twitter client types. At night time, an overwhelming majority of people use mobile Twitter clients to send their responses (Figure 3(c)). Overall, mobile client users consistently dominate over desktop/laptop users (Figure 3(c)).

In the second experiment, we compare the user responses from different cities. We observe that users in NYC respond quicker than those in Toronto and Montreal, which have almost the same response patterns (Figure 4(a)). In Figure 4b, we compare the participation ratio of the users in these three cities. We see that users in NYC participate more than those in Toronto and Montreal (Figure 4(b)). In all these three cities, mobile Twitter client users dominate over desktop/laptop users and this ratio is highest in NYC (Figure 4(c)).

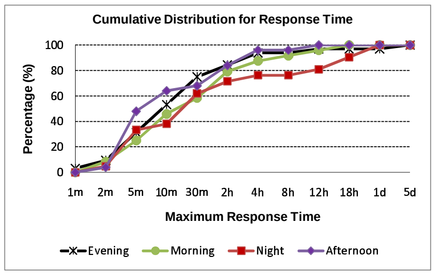
TABLE I
COMPARISON OF USER RESPONSES WITH WEATHER.COM

City	Match for Current Day	Match for Next Day
New York City	89%	56%
Toronto	79%	29%
Montreal	88%	54%

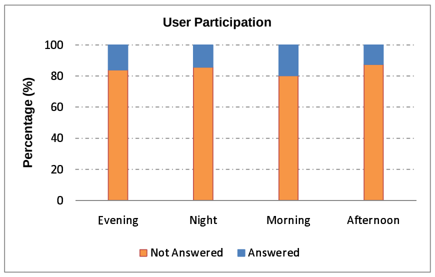
In the final experiment, we analyze the correlation of answers from our users with data from Weather.com. Since it is not practical to validate Twitter user responses with various fine-grain spatial (latitude, longitude) and temporal dimensions, the correlation is based on course-grain city wide level weather data for the entire day.

In the first column of Table I, we list the correlation of user responses with the data from weather.com for the current day (the weather.com data and user responses are collected in the same day). If the weather.com reports "snowy" for the day, all responses except "snowy" are counted as "unmatched". If the weather.com reports a fuzzy condition such as "partly cloudy", all responses including "sunny" and "cloudy" are counted as "matched". In this experiment, we observe that for each city at least 79% of the answers match with the data from weather.com.

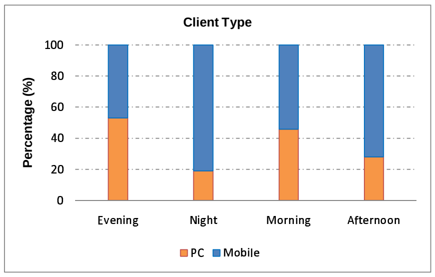
In the second column of Table I, we list the correlation of user predictions for the next day with the data from weather.com. Here we collect the predictions of users in



(a) CDF for response time

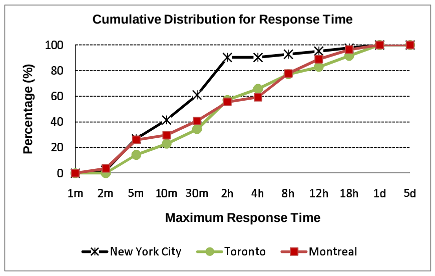


(b) User participation

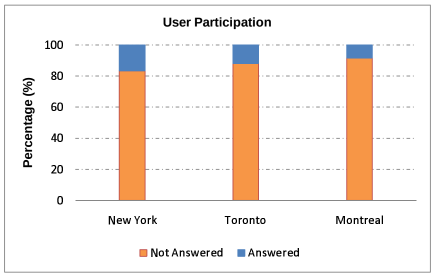


(c) Client types

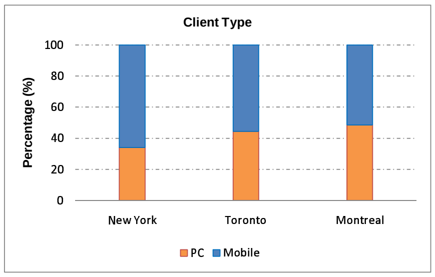
Fig. 3. Experimental results for NYC in different time slices



(a) CDF for response time



(b) User participation



(c) Client types

Fig. 4. Experimental results for 3 cities

previous day (December 6) and find the correlations of those predictions with weather.com data collected on the next day (December 7). We observe that at least 50% of the user predictions match with weather.com for New York City and Montreal whereas it is 29% in Toronto.

VI. Case Study: Smartphone Enabled Noise Map

In this application, we measure the noise level of the surrounding environment via GPS enabled smartphones and provide a noise level querying service over Twitter. We describe our implementations of the Askweet and Sensweet components for this application below.

Askweet component. We implemented the Askweet component similar to that of the weather radar application. The noise map application has its own query format of "?Noise Loc:Location". Any Twitter user can send a question to the Twitter account of Askweet (twitter.com/askweet) in order to query the noise level of a specific location. For example "?Noise Loc:Student Union, UB, Buffalo, NY" queries for the noise level of the Student Union at the University at Buffalo.

When Askweet gets a new query, it automatically tries to resolve the location by using Google's Geocoding Service (http://code.google.com/apis/maps/documentation/). After getting the latitude and longitude information from Google's Geocoding Service, Askweet searches previously known Sensweet clients in the database in proximity of the specified location. If Askweet finds a local client, it returns the latest noise level obtained from that client. If multiple Sensweet clients are found, the noise value with the latest timestamp is returned to the querier.

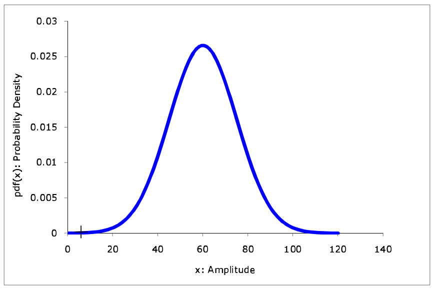
Sensweet component. We implemented a Sensweet client for the Nokia N97 Smartphone series. For implementing the Sensweet client we used Carbide C++ version 2.0.2, Nokia N97 Symbian S60 SDK V1.0 and Qt Tower 4.5.2. The total size of the source code for this Sensweet component is more than 1500 lines of code.

The Sensweet client detects the noise level of the surrounding environment and forwards this data to Twitter using our TweetML format mentioned in Section IV-B. The specific TweetML format ($|Loc|Noise : Val|Timestamp|$) for Noise Map application includes ordered values for location, sensor reading and timestamp. An example sensor reading can be "Noise:H" denoting that the current noise reading is "High". Since Nokia N97 smartphones do not provide the noise level in decibel format, we implemented our own noise sensor driver to map noise samples into three categories: L as Low, M as Medium and H as High.

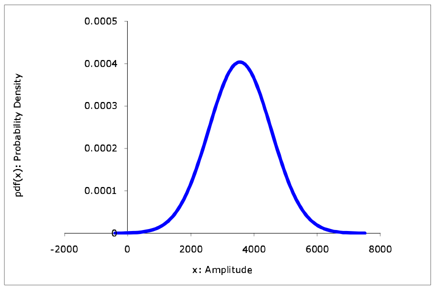
Our Sensweet client implements a timer for reading the GPS coordinates and using the microphone to record a one second noise sample in "Windows WAV" file format. Then, Sensweet client parses this WAV file to obtain the mean value for the amplitude of signals in the sample. In order to map the current sample into one of the noise categories {Low, Medium, High}, we used three normal distributions. For a given mean value x of amplitudes obtained from a one second sample, we calculate the following probability density function (pdf(x)) for each of the predefined 3 normal distributions:

$$pdf(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{(x-\mu)^2}{2\sigma^2}) \quad (1)$$

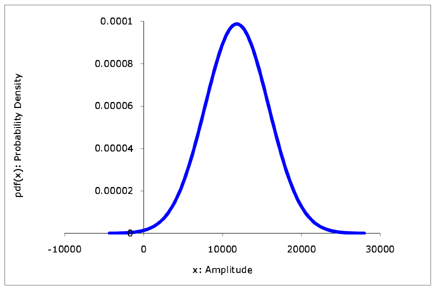
The μ in the formula represents the mean of the corre-



(a) Low level noise

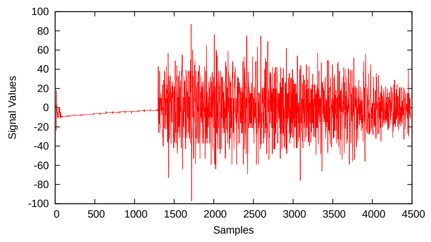


(b) Medium level noise

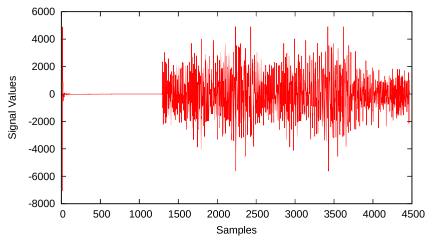


(c) High level noise

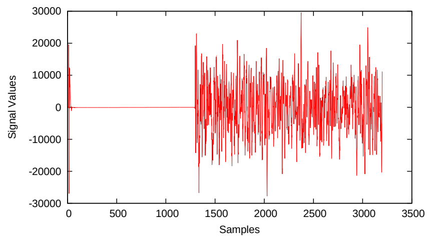
Fig. 5. Normal distributions for different noise levels



(a) Low level noise



(b) Medium level noise



(c) High level noise

Fig. 6. Representative samples for different noise levels

sponding distribution and σ^2 represents the variance. The assignment is based on the highest value. Since there is no gain setting for the microphone of Nokia N97, our mapping is valid for any Nokia N97 smartphone device. For the smartphones having adjustable microphone gain, our mapping can be easily adapted by dividing signal values by the gain factor.

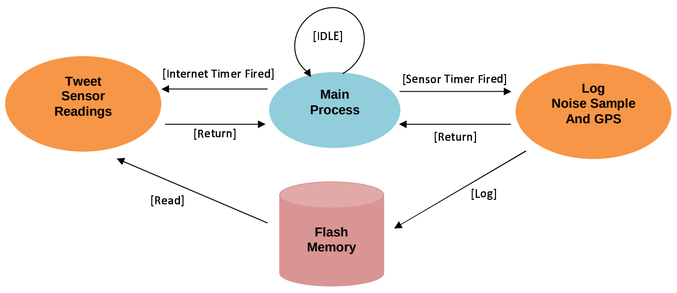


Fig. 7. State transition diagram for Sensweet client

The state diagram of the Sensweet client for noise map application is given in Figure7. When the phone is started the Sensweet application is also launched as a background process and waits in the "idle" state. The GPS based location, noise level, and current timestamp is logged to the flash memory when the sensor timer is fired. We also keep another timer for forwarding sensor readings to Twitter. When the Internet timer is fired, main application reads the latest sensor readings from the flash disk and tweets it (http://twitter.com/Sensweet).

A. Experiment Results

Here we provide our experimental results for the noise map application.

In order to determine the normal distributions representing the "Low", "Medium", and "High" categories for noise levels, we performed experiments in six different locations with varying noise levels. In each location, we recorded more than 200 noise samples with a duration of one second.

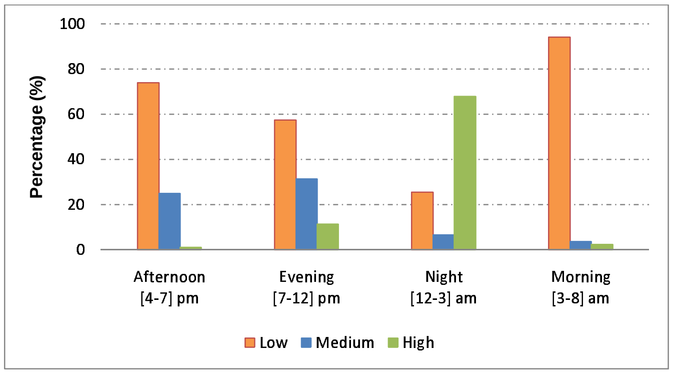


Fig. 8. Daily noise fluctuation graph

We assign the "Low" category to the samples that we obtained during the silence in home and computer lab locations. The amplitude distribution for "Low" level noise is given in Figure 5(a). Here the amplitude (absolute value of signal values) of low level noise mostly fluctuates between [0,100], which also implies that signal values mostly fluctuate between [-100,100] (Figure 6(a)). For the "Medium" category we collect samples from the Student Union at UB and various meeting rooms at the CSE department where people talk to each other (noise mostly includes human voice). The "High" category is collected in bars and clubs in Buffalo with loud background music. The normal distribution of amplitudes for

"Medium" and "High" categories are given in Figure 5(b) and Figure 5(c). Representative samples for these two categories are also given in Figure 6(b) and Figure 6(c) respectively.

In another experiment, we measure the noise fluctuation of our case study user for one weekend day over different time slices starting from Saturday 4.00 pm until Sunday 8.00 am (Figure 8). By analyzing the temporal noise fluctuation, it can be possible to predict some of the activities of the user during the day time. In the afternoon period the noise level fluctuates between "Low" and "Medium" level. During this time the user was at home and meeting with his friends. In the evening period the ratio of "Low" level decreases and ratio of other two levels increase. In this period, the user was having dinner with his/her friends in some place and going to a bar/club after that. In the night period the noise level is mostly "High" and the user was visiting a club. The noise level in the morning period is "Low" mostly since the case user was sleeping at home.

VII. Concluding Remarks

We presented a crowd-sourcing system architecture over Twitter, and demonstrated this system with two case studies: weather radar and noise mapping. Our experiments with crowd-sourcing on Twitter are promising. Even without an incentive structure, Twitter users volunteer to participate in our crowd-sourcing experiments (with around 15% reply rates) and the latency of the replies are low (50% replies arrive in 30 minutes and 80% replies arrive in 2 hours). Another promising finding is that a majority of replies were tweeted from smartphones.

Our experiments suggest that Twitter provides a suitable open publish-subscribe infrastructure for tasking/utilizing sensors and smartphones and can pave the way for ubiquitous crowd-sourced sensing and social collaboration applications. There are several open research questions remaining for fulfilling this vision. Security and trust issues remain as significant challenges. In our future work we will consider mining of tweets and exploiting of social networks structures in Twitter to deploy expert finding and social collaboration applications. We will also experiment with adding various incentive schemes to our crowd-sourcing system, caching replies, and deploying our system on a cloud computing platform.

References

[1] M. Weiser, "The future of ubiquitous computing on campus," *Commun. ACM*, vol. 41, no. 1, pp. 41–42, 1998.

[2] N. Eagle and A. Pentland, "Social serendipity: Mobilizing social software," *IEEE Pervasive Computing*, vol. 04-2, pp. 28–34, 2005.

[3] "www.twitter.com."

[4] "State of twittersphere," http://cdnqa.hubteam.com/State_of_the_Twittersphere_by_HubSpot_Q4-2008.pdf, December 2008.

[5] "State of twittersphere," http://blog.hubspot.com/Portals/249/sotwitter09.pdf, June 2009.

[6] "http://twitter.grader.com/top/cities."

[7] "https://www.technologyreview.com/files/18810/forward.pdf," October 2008.

[8] "http://www.washingtonpost.com/wp-dyn/content/article/2009/06/16 /ar2009061603391.html," June 2009.

[9] "http://twitpic.com/135xa," January 2009.

[10] "http://www.economist.com/world/middleeast-africa/displaystory.cfm?story_id=13856224," June 2009.

[11] "http://www.maine.gov/portal/cas/index.shtml."

[12] "http://www.e2campus.com/pr081203-pacific_facebook_twitter.htm," December 2008.

[13] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, Y.-R. Choi, T. Herman, S. S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora, and M. Miyashita, "A line in the sand: A wireless sensor network for target detection, classification, and tracking," *Computer Networks (Elsevier)*, vol. 46, no. 5, pp. 605–634, 2004.

[14] A. Arora, R. Ramnath, E. Ertin, P. Sinha, S. Bapat, V. Naik, V. Kulathumani, H. Zhang, H. Cao, M. Sridharan, S. Kumar, N. Seddon, C. Anderson, T. Herman, N. Trivedi, C. Zhang, M. Nesterenko, R. Shah, S. Kulkarni, M. Aramugam, L. Wang, M. Gouda, Y. Choi, D. Culler, P. Dutta, C. Sharp, G. Tolle, M. Grimmer, B. Ferriera, and K. Parker, "Exscal: Elements of an extreme scale wireless sensor network," *Int. Conf. on Embedded and Real-Time Computing Systems and Applications*, 2005.

[15] R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. M. Mainwaring, and D. Estrin, "Habitat monitoring with sensor networks," *Commun. ACM*, vol. 47, no. 6, pp. 34–40, 2004.

[16] G. Werner-Allen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh, "Fidelity and yield in a volcano monitoring sensor network," in *in 7th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2006)*, 2006.

[17] "www.twitterpeek.com."

[18] J. Burke and et al., "Participatory sensing," in *ACM Sensys World Sensor Web Workshop*, 2006.

[19] D. Metzler, S. Dumais, and C. Meek, "Similarity measures for short segments of text," in *In Proceedings Of ECIR-07*, 2007.

[20] V. Hatzivassiloglou, J. L. Klavans, and E. Eskin, "Detecting text similarity over short passages: Exploring linguistic feature combinations via machine learning," in *In Proceedings Of The 1999 Joint SIGDAT Conference On Empirical Methods In Natural Language Processing And Very Large CORPORA*, 1999, pp. 203–212.

[21] K. Balog, L. Azzopardi, and M. de Rijke, "A language modeling framework for expert finding," *Inf. Process. Manage.*, vol. 45, no. 1, pp. 1–19, 2009.

[22] Q. Mei and C. Zhai, "Discovering evolutionary theme patterns from text: an exploration of temporal text mining," in *KDD '05: Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. New York, NY, USA: ACM Press, 2005, pp. 198–207. [Online]. Available: http://dx.doi.org/10.1145/1081870.1081895

[23] J. Sankaranarayanan, H. Samet, B. E. Teitler, M. Lieberman, and J. Sperling, "Twitterstand: news in tweets," in *GIS '09: Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. New York, NY, USA: ACM, 2009, pp. 42–51.

[24] T. Nasukawa and J. Yi, "Sentiment analysis: capturing favorability using natural language processing," in *K-CAP '03: Proceedings of the 2nd international conference on Knowledge capture*. New York, NY, USA: ACM, 2003, pp. 70–77.