Integrity of Data in a Mobile Crowdsensing Campaign: A Case Study

Heng Zhang, Saurabh Bagchi, He Wang
Purdue University, West Lafayette, IN, 47907, USA,
[zhan2614,sbagchi,hw]@purdue.edu

ABSTRACT

Mobile crowdsensing (MCS) has a huge potential to provide societal benefits by effectively utilizing the sensing, computing, and networking capability of mobile devices, which have become ubiquitous especially in the developed world. However, many challenges come along with MCS such as energy cost, privacy issues, and the data integrity of crowdsensed data. In this paper, we focus on analyzing the data integrity of mobile crowdsensed data in a user study of 60 people who participate in a crowdsensing campaign that collects barometric pressure data from various locations of our campus. Each set of 20 users runs one of the three different crowdsensing frameworks, one of which, called Sense-Aid, is an energy-efficient framework designed by us. We analyze the characteristics of the data with respect to their integrity. From our analysis, we find that for 90 MCS tasks there is a surprisingly high number of outlier percentage (close to 20%), only about 10% of the participants report data for the entire duration of 7 days of the MCS campaign, and using a reputation system to filter out spurious data values helps in getting within 0.8% of the ground truth barometric pressure, whereas not using the reputation system leads to a discrepancy of about 20%. We hope our analysis will provide a useful reference to MCS researchers and developers in their future work running MCS campaigns.

CCS CONCEPTS

• **Networks** → *Network performance evaluation*; Network architectures;

KEYWORDS

Mobile Crowdsensing, Data integrity, Data analysis

ACM Reference Format:
Heng Zhang, Saurabh Bagchi, He Wang. 2017. Integrity of Data in a Mobile Crowdsensing Campaign: A Case Study. In *Proceedings of First ACM Workshop on Mobile Crowdsensing Systems and Applications (CrowdSenSys'17).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3139243.3139255

1 INTRODUCTION

Mobile Crowdsensing (MCS) refers to an activity where data is sensed and generated by multitudes of people using devices that they carry with them, typically mobile cellular devices, and the data is consumed by a MCS server. The MCS server synthesizes the sensed data to determine some information of relevance to the MCS campaign organizers. MCS has become a popular way to collect information about some phenomenon that happens over a period of time or a spread out geographical region (or both). For example, the data contributed from participants can be used to improve noise pollution [20], or to provide gas price information [5]. There are two main categories in MCS which are participatory crowdsensing and opportunistic crowdsensing. The former requires a user's active involvement such us taking a photo at a specific location and sending it to server. The latter relies only on the devices without a user's active involvement. For example, an MCS application for determining noise pollution level can periodically sample the noise level through the microphone sensor. MCS is beneficial in many areas because of its flexibility and wide utilization of the powerful resources on millions of mobile devices to collaboratively serve a common purpose. For example, to broadcast a real-time and reliable weather map of an approaching tornado, rather than relying on traditional air pressure measurement stations across the city, MCS can be used by sampling the barometric reading from barometer sensors on smartphones. However, many challenges exist in designing an MCS campaign and they have to be addressed before MCS can be widely adapted into our daily lives. For example, ensuring the integrity of the collected data is a big challenge. During an MCS campaign, the data provided by users may be misleading such as intentionally falsified data, hardware/software failure, network issues. One possible source of error that may crop up is the lack of synchronization among the client devices and the server infrastructure. However, we can use low-duty synchronization protocols such as [14] to avoid this source of error. There are other challenges such as user privacy, energy cost, users' incentives. To motivate more participants and to provide confidence in the results of an MCS campaign, these challenges have to be resolved.

Our focus is data integrity challenge in a MCS campaign. A variety of factors that threaten data integrity in an MCS activity are discussed in [8, 9]. Because MCS requires the involvement of unpredictable participants, the MCS server can receive erroneous or malicious data input. Intentionally, MCS participants may submit fake data. For example, in a transportation MCS campaign, participants may report wrong traffic condition information because they do not want other people to choose a route that goes near their homes. Unintentionally, there may be environmental reasons such as uncertainty in GPS location when participants move indoors. There are other natural reasons leading to faulty data, such as hardware failures on mobile sensing devices, network congestion affecting data transmission, noise in sensed data, etc. Besides even if each participant contributes accurate data, the overall aggregated

CrowdSenSys'17, November 5, 2017, Delft, The Netherlands

ACM ISBN 978-1-4503-5555-1/17/11…$15.00
https://doi.org/10.1145/3139243.3139255

data may not be accurate because of lack of enough temporal or spatial granularity.

In this paper, we describe a user study for an MCS campaign that we conducted on our campus and the data integrity issues that arose. The user study was conducted to evaluate the energy efficiency of Sense-Aid, a MCS energy-saving framework proposed by us. Figure 1 shows the architecture of it and details are described in Section 2.2. Specifically, we look into the three characteristics of MCS data related to integrity. *First*, what is the proportion of outlier data among all the measurements. *Second*, what is the extent of participation by different users over the duration of the 7-day campaign. *Third*, can a reputation system be used at the MCS server to increase the integrity of the synthesized data. Specifically, our study invited 60 students to go about their daily business on campus and use the barometer sensor on their smartphones to collect and send atmospheric pressure data. We divide the volunteers into 3 groups, corresponding to 3 different MCS frameworks, Sense-Aid, Piggyback Crowdsensing [15], and a baseline Periodic framework, which simply uploads data periodically.

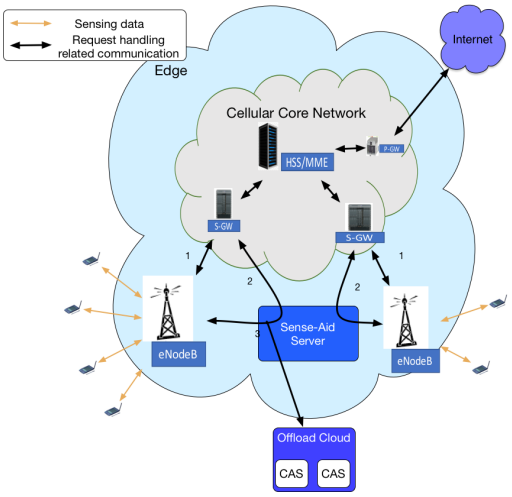

Figure 1: Sense-Aid deployed on a cellular LTE network, showing the three major components: Sense-Aid server, Sense-Aid clients, and crowdsensing application server (CAS). eNodeBs communicate with the cellular core network through path 2 if its incoming traffic from UEs includes crowdsensing data. Otherwise, eNodeBs choose path 1.

There are some prior works [2, 6, 7, 11, 16, 19] that have tackled the data integrity problem in the face of unreliable users or unreliable environment. One approach uses a reputation system to evaluate each participant's sensed value and adjusts that score every time the participant reports MCS data. However, this approach cannot fulfill different types of the MCS campaigns. There may not exist a generic way to cover all types of MCS activities and provide the data integrity guarantee nor is this paper aimed at discussing such a generic approach. Rather we adopt the reputation systems from [10] to show the importance and benefits reputation system.

The rest of the paper is organized as follows. Section 2 talks about the reputation system and gives background of the Sense-Aid framework. Section 3.1 gives the details of our user study. Characteristics of MCS data and the effect of reputation system are evaluated in Section 4. Section 5 gives our conclusions and potential future work to improve MCS data integrity.

2 BACKGROUND

2.1 Reputation System

A reputation system is used to help filter out unreliable participants and reduce the noise that affects data integrity in an MCS campaign. The state-of-the-art Beta reputation system [13], for example, uses beta probability density functions to combine feedback and derive reputation ratings. The crowdsensing server will maintain reputation scores for all participants and keep updating each participant's score anytime that participant sends back crowdsensed data. A high reputation score means that data from that participant has been reliable in the past and the server will tend to weigh that participant's contribution higher in synthesizing the final output. We choose to adopt the reputation system from [10] for our user study. It has two modules: a watchdog module and a synthesis module. The watchdog module generates a confidence score between 0 and 1 from the readings submitted by a user. Confidence score is defined as how confident the MCS server is to choose the data sent from that participant. The input of watchdog module is a vector of sensor values. The confidence score is computed in an iterative way for each data upload as shown in Algo. 1. The convergence relies on progressive identification and elimination of outlier data points. There are two categories of outlier detection algorithms. One is model-based [19] and the other is consensus-based [7]. Model-based outlier detection algorithm relies on the prior knowledge of the physical process that is being sensed, such as, how the amplitude falls off with distance. The consensus-based outlier detection approach reaches consensus from the data sent in by a group of nodes and eliminates outlier data points. In our evaluation, we take the consensus-based approach to detect outliers because there is a natural density of nodes (our study participants) sensing the same physical measure (atmospheric pressure) in geographic proximity.

The second module of the reputation system, the synthesis module is fundamentally a function to generate the reputation score, based on each participant's confidence score from watchdog module. The reason that the confidence score is not directly used as user's reputation is that it has been found useful for the reputation score to have some properties [7, 10]. The first property is a smooth bonus growth with correct behavior and the second property is a severe penalty upon inaccurate data upload. The general idea of these properties is that if a participant contributes a trustworthy value, its weight will gradually increase but if the contribution is faulty data, its weight will significantly drop because failure behavior tends to be bursty either because of environmental constraints (*e.g.*, the participant is in a GPS-starved region) or user motivation (*e.g.*, the user is incentivized to provide inaccurate data). We use the Gompertz function to provide the smooth-bonus property while the severe-penalty property is achieved by quadratically downgrading the reputation score. Gompertz function is a mathematical model for time series which has slow growth rate at the start and end time period as well as a gradual growth rate in between. The representation of Gompertz function is as follows:

$$R_{(i)}(p'_{(i)}) = a \exp^{b \exp^{cp'_{(i)}}} \quad (1)$$

where $R_{(i)}$ is the reputation score, i is the participant index, $p'_{(i)}$ is the normalized confidence score for participant i (normalization is discussed in Section 3.1), a, b, c are control variables that represent upper asymptote, X-axis displacement, and growth rate respectively. An illustrative plot of Gompertz functions with different choices of control variables is shown in Figure 2. We choose $a = 1$ to maintain a [0,1] reputation score range, $b = -2.5$ to make sure the X-axis is not too detoured from the origin, and $c = -0.85$ to have a smooth growth rate.

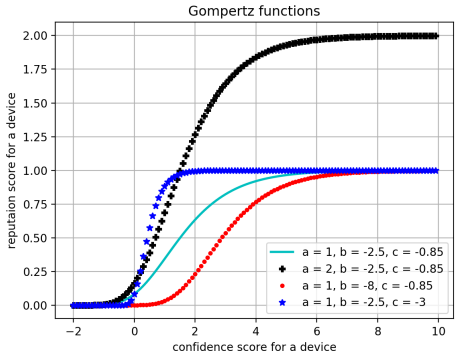

Figure 2: Gompertz function with different control variables

2.2 Sense-Aid: Energy-efficient MCS framework

Sense-Aid is an MCS framework to save mobile device energy consumption. It achieves this goal through cooperation between the cellular network and the mobile device, with parts of its framework code installed at these two elements. There are three key innovations in Sense-Aid. First, It runs an orchestration and scheduling algorithm on the cellular network edge to select the appropriate devices to task with uploading data at any point in time. It does this so as to suppress redundant data collection and communication. Second, it understands the state of each device (radio state, battery state, etc.) and this helps it to decide which devices should be selected for crowdsensing activities at any point in time. It can thus ensure that there are enough devices queried to satisfy the temporal and spatial requirement, while not causing excessive energy drain on any one device and ensuring some degree of fairness of usage across devices. Finally, Sense-Aid has a predictive component to estimate when some devices will be in a geographical location so as to be tasked in the near future. It can therefore delay a data collection for some bounded time period when there is the possibility of a near-future, high-fidelity MCS data collection. The schematic architecture for Sense-Aid is shown in Figure 1. The Sense-Aid server resides in the cellular edge while the crowdsensing application server (CAS) can exist on any computing environment, such as, a cloud computing environment. The different cellular devices act as the clients in the MCS campaign.

Previous work from Carrol *et al.* [1] has shown that the majority of power consumption on mobile devices can be attributed to the GSM module and the display. We posit that if the crowdsensing task were made aware of the state of the radio, it could do its job in a more energy efficient manner. For example, the framework can prioritize a device whose radio is already active over a device whose radio is in sleep state, considering that there is a big energy spike in moving from the sleep to the active state.

3 USER STUDY

3.1 Setup

The user study involved creating an atmospheric pressure map of parts of the campus, using barometer readings from smartphones of multiple volunteer student participants. The MCS campaign lasted for one week with 8 hours each day (8 am-4 pm). We have 60 students on our campus participate in the user study. The crowdsensing tasks have parameters explained in Table 1. There are 4 test locations on our campus, *Student Union (Loc1), EE department (Loc2), CS department (Loc3), and University Gym (Loc4).* Their geographic locations are shown in Figure 3. We conducted 4 experiments as summarized in Table 2. Each experiment consists of multiple tests, where one parameter is varied to go from one experiment to the next. For example, experiment 1 has 6 tests where area radius for data collection at each location is varied. Using Sense-Aid, it is possible for the crowdsensing server to assign one or more tasks to run concurrently on each device. For example, there may be two different crowdsensing campaigns, one creating the noise map in the city and another measuring bumps on the streets and one user with one smartphone could be concurrently running both tasks. In fact, our experiment 4 explicitly measures the effect of multiple tasks running on one device. We promise as incentive to the users $25 Amazon gift card at the end of the 7 day period of the campaign. Implicitly if a participant uploads any data during the course of the campaign, she would receive the incentive payment. The 60 users are divided into 3 groups of 20 users each. Every group of users participates in the same MCS campaign but run different MCS client frameworks. The three MCS frameworks are Periodic (as a baseline), Piggyback Crowdsensing (as the prior best energy-efficient MCS framework), and Sense-Aid. The user study was designed to compare the energy efficiency of the three frameworks but here we only focus on the data integrity issue and therefore do not delve into the energy saving aspects of the frameworks.

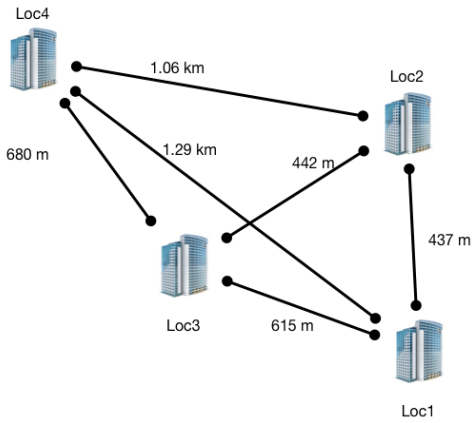

Figure 3: User study target buildings

3.2 Reputation system in our user study

We simplify the reputation system from Huang, *et al.* [10] and use it in our user study to synthesize the output from the collected

Parameter	Details
int sampling_period	Specifies the period between two consecutive samples. e.g. sampling_period of 5 secs implies one sample every 5 seconds.
float area_radius	Given a specific location by crowdsensing task the radius determines the area for Sense-Aid server to select devices.
int spatial_density	Represents the number of devices required in the area specified above.

Table 1: Task parameters.

Experiment Number	Varying parameter	Default parameters
Experiment 1	Area radius 100 m, 200 m, 300 m, 400 m, 500 m, 1000 m	test duration = 1.5 hrs/test number of tasks/device = 1 sampling period = 10 mins spatial density = 2
Experiment 2	Sampling period (1, 5, 10) [min]	test duration = 2 hrs/test number of tasks/device = 1 spatial density = 3 area radius = 500 m
Experiment 3	Spatial_density (1, 2, 5, 7)	test duration = 1.5 hrs/test number of tasks/device = 1 sampling period = 5 mins area radius = 1000 m
Experiment 4	Number of tasks per device (3, 5, 10, 15)	test duration = 1.5 hrs/test sampling period = 5 mins spatial density = 3 area radius = 500 m

Table 2: Experimental parameter settings in user study and summary energy saving results.

readings. The reputation system has the watchdog module feeding into the synthesis module (the latter is called the "reputation system" in the paper but we change the name to eliminate the ambiguity).

Watchdog module

The watchdog module is used to calculate the confidence score. It is computed in an iterative way and its pseudo code is shown in Algo. 1. The intuition behind Eq.2 and Eq.3 is the robustness provided by combing these two equations as proved by Chou *et al.* [3]. When applying the watchdog module to the user study, each of the 60 users will (ideally, in an error free environment) submit a single barometric value at one time slot for one location. We use i as the device index, t as time slot index where it goes up from 1 till the duration of the test, as laid out in the experiment configuration shown in Table 2. We define the average sensor value in time slot t as:

$$a_t = \sum_{i=1}^{n} p_i x_{i,t} \tag{2}$$

where p_i is the confidence value for device i computed from Eq. 3. Initially it is set to $1/n$ where n is the total number devices. With the help of Eq. 2, we can polish the confidence score using the following function borrowed from Huang *et al.* [10] which drives this function from Chou *et al.* [3].

$$p_i = \frac{\frac{1}{\frac{\sum_{t=1}^{T}(x_{i,t}-a_t)^2}{\sum_{i=1}^{n}\sum_{t=1}^{T}(x_{i,t}-a_t)^2}+\epsilon}}{\sum_{p=1}^{n}\frac{1}{\sum_{t=1}^{T}(x_{p,t}-a_t)^2}} + \epsilon \tag{3}$$

where ϵ is convergence threshold. The server keeps running Eq. 2 and Eq. 3, and update p_i until the convergence threshold is reached. After that, the watchdog module passes the confidence score to the synthesis module.

Algorithm 1 Watchdog Pseudo Code

1: **procedure** Watchdog Module
2: ▷ l stands for lth iteration. i is device index. p is confidence score.
3: Initialization: $l = 0$; $p_i^0 = 1/n$;
4: Calculate a_t^{l+1} as shown in Eqn. 2;
5: Calculate p_i^{l+1} as shown in Eqn. 3;
6: Increment l by 1 and repeat step 2 to 4 if no convergence, which is defined as $|p^{l+1} - p^l| < \epsilon$

Synthesis module

We use the Gompertz function for the synthesis module for reasons as explained in Section 2.1. We choose a, b, and c to be 1, -2.5, and -0.85. a is the upper asymptote and we want to limit the score within [0,1].

$$R_i(p_{(i)}) = \exp^{-2.5\exp^{-0.85p_i}} \tag{4}$$

The choice of a, b, and c shifts the zero reputation score approximately to the origin. Note that a missing value does not bring down a participant's reputation but only an inaccurate value does. In the end, to properly use the synthesis module, we need to normalize the confidence score so that the input in mapped to proper interval of Gompertz function. We normalize the output of the watchdog module so that p_i lies within [-2, 2]. This is done through Eqn. 5. The interval is chosen based on Figure 2 when $a = 1$, $b = -2.5$ and $c = -0.85$. If a device's confidence score is lower than 0, which means the server has less confidence of the integrity of the contribution from this user, the growth rate is slow while as the confidence increases, the growth rate gradually increases.

$$p'_{(i)} = \frac{4 \times (p_i - min\{p_i\}_i^n)}{max\{p_i\}_i^n - min\{p_i\}_i^n} - 2 \tag{5}$$

4 EVALUATION

For each of the 4 experiments in our user study, we have several tests and for each test, we have multiple tasks running concurrently on the MCS backend server. For this user study, each test results in 4 tasks at the server, corresponding to the 4 locations. To show the result, we aggregate the data for three frameworks because with respect to data integrity, they pose no difference. In total, we have 90 tasks assigned to 60 MCS clients in the one week of our study.

4.1 Outlier Analysis

Figure 4 shows the outlier percentage for each task. We user Tukey's criteria to define outlier with the choice of k to be 1.5 [21]. We use

the following formula to determine the outliers.

$$[Q_1 - k(Q_3 - Q_1, Q_3 + k(Q_3 - Q_1)) \quad (6)$$

where Q_1 and Q_3 stands for first quartile and third quartile, $k = 1.5$ was proposed by John Turkey [21] which defines the fences for outlier. We see that across all the tasks, the average percentage of outliers is 7.2%, a number that is higher than what we had originally expected. It means a significant number of users have bad or missing data in a typical MCS campaign and their contributions need to be filtered out. The bad or missing data can happen due to inaccuracies in the sensor (they have a setting and initialization time), the user being out of coverage range, or the phone being switched off. In further analysis (not reported here), we distinguish between bad and missing data.

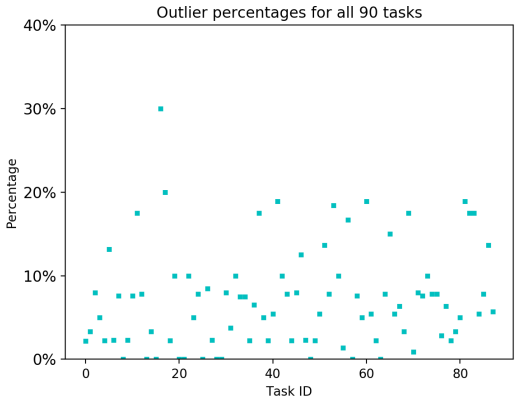

Figure 4: Outlier Percentage per task. The outliers are filtered out by Formula 6. We compare the reputation scores for all participants in one task to detect the outliers.

4.2 Participation Analysis

Figure 5 shows the participation of users with respect to the number of days each user participated in. Since the lack of participation across users did not line up on the same days, for any given day we had enough data points. The definition of participation in a day for a user is that the user contributed at least one non-zero value for that day. We can clearly see the jump of 1 day's participation and 2 days' participation. This is understandable and the message behind this drain is that after participants are invited the first time, they are curious and willing to participate but after the novelty wears off, some people lose interest and leave the MCS campaign. We can see that only a small fraction (about 10%) of the users participate throughout the duration of the campaign, even though the duration of the overall campaign is quite short (7 days). To address the low participation, one may set up the incentive system differently, such as, to take into account participation in each day, and even further, by the fidelity of the data contributed by the user. Such incentive design has been discussed in [12] and our case study bears out these theoretical proposals.

4.3 Benefits of reputation system

We choose 12 tasks' MCS data to analyze the reputation system in one day's participation. This raw data is part of the data collected

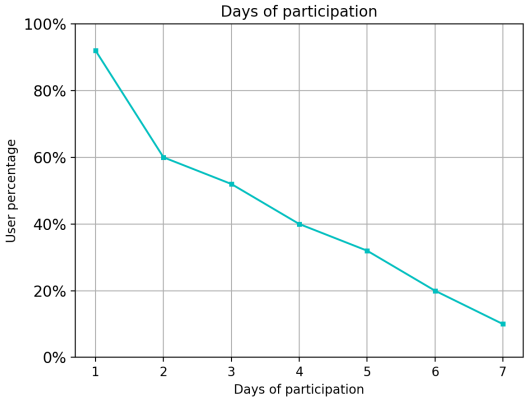

Figure 5: Drain of Participation with time. Even with the the monetary bonus, users' incentive drains as time passes by. Only about 10% of the users participate during the entire 7 days of the user study.

in Experiment 1 in Table 2. Each task lasts for 90 minutes and since sampling period is 10 minutes, each participant submits 9 barometric values for one task. For the four tasks which require 1000 m area radius, there are 38 clients. This number is 21 and 18 for radius 500 m and 400 m respectively. This decreasing number is expected since the radius is reduced and less users are present in the required area. As in Figure 6, we see that there are always some people whose reputation score are lower that the bar line in the figure. That bar line is calculated as $1/n$ where n is the number of participants in that test. Note that if a participant contributes one zero value, its reputation score is reset be Gompertz ($1/n^2$). Therefore, it needs several correct contributions to build up its reputation. That is why we see a great variability in the plot. Those that stay below the bar line are contributing many zeroes while those staying toward the top are almost always uploading trustworthy data. Those that stay in the middle are participants who occasionally send zero after which they send trustworthy data for some time and build up their scores. We call the participants with a reputation score higher than the bar line as preferred-group. When the number of people in preferred-group cannot satisfy the MCS task, MCS server chooses from the group of people whose reputation scores are lower than the bar line. With the help of the reputation system, the MCS campaign result will become more accurate and reliable over time.

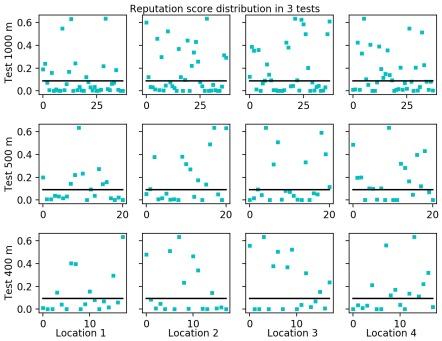

Figure 6: Reputation scores in 3 tests

4.4 Fine-grained user data Analysis

We look into two types of granularity. Figure 7 shows 5 examples of weekly barometric values that collected by the users in preferred group. We add the real weekly barometric value in our city from Wunderground [4], a weather forecast website. The trend of barometric value from the selected users by the reputation system is similar to that of the real value. The gap between real value and collected value on each day is because of distance between our campus and the measurement station used in Wunderground. They are 2.6 miles and roughly 20 meters difference in altitude. Therefore, this difference of 0.8% in the pressure reading can be expected.

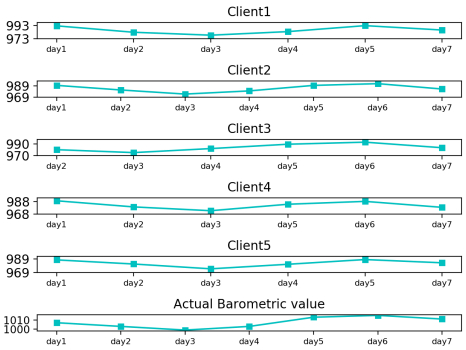

Figure 7: Comparison of reliable users' uploaded data with the actual daily barometric values

To show the benefits of the reputation system, we compare average barometric values between combining all users and combining those from the preferred-group users from the first six days. The sample pool for the seventh day's data is too small (10%) so we do not add it here. We see in Figure 8 that after filtering out disreputable users' contributions using the reputation system, the average daily barometric values are within 0.8% of the ground truth barometric pressure but if we remove the reputation system, the discrepancy increases to about 20% of the ground truth. Therefore, in an MCS campaign, a reputation system can positively assist the MCS server to draw accurate results from the uploading users' raw data.

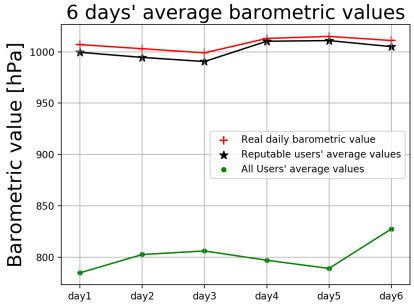

Figure 8: Effect of the reputation system - without filtering, the MCS aggregated data deviates significantly from the "ground truth".

5 RELATED WORK AND CONCLUSION

We believe that in most MCS campaigns, unreliable data will be a fact of life and it is crucial to filter out unreliable data to achieve accurate results. However, to ensure the data integrity in an MCS campaign is not an easy job because of the complexity of the MCS campaign and various uncertainties due to the environment, the network status, the user device, and the motivation of the users. We see that in our MCS campaign involving 60 participants over a week, there is a non-trivial amount of spurious or missing data. We see that using a reputation system to identify and then filter out spurious data gets the crowdsensed information to be much closer to the ground truth. There are also other methods to solve this problem such as truth discovery approach by Meng, *et al.* [17, 18]. When we were running the reputation system to calculate users scores, it takes approximately 20 seconds to deliver the results. In our limited scale MCS campaign, this latency is tolerable. However, we envisage that the latency will increase with the number of participants and the latency may become intolerable for some campaigns. This points to the importance of dedicating enough computation resources for ensuring the high integrity of sensed data.

REFERENCES

[1] A. Carroll and G. Heiser. 2010. An Analysis of Power Consumption in a Smartphone.. In *ATC*. 1–14.
[2] D. Christin, C. Roßkopf, M. Hollick, L. A Martucci, and S. S Kanhere. 2013. Incognisense: An anonymity-preserving reputation framework for participatory sensing applications. *Pervasive and mobile Computing* 9, 3 (2013), 353–371.
[3] C. T. Chun, A. Ignjatovic, and W. Hu. 2013. Efficient computation of robust average of compressive sensing data in wireless sensor networks in the presence of sensor faults. *TPDS* 24, 8 (2013), 1525–1534.
[4] The Weather Company. 2017. Wunderground. https://www.wunderground.com. (2017).
[5] Y. F Dong, S. Kanhere, C. T. Chou, and N. Bulusu. 2008. Automatic collection of fuel prices from a network of mobile cameras. *Lecture Notes in Computer Science* 5067 (2008), 140–156.
[6] A. Dua, N. Bulusu, W. Feng, and W. Hu. 2009. Towards trustworthy participatory sensing. In *HotSec*. 1–6.
[7] B. Ganeriwal, L. K Balzano, and M. B Srivastava. 2008. Reputation-based framework for high integrity sensor networks. *TOSN* 4, 3 (2008), 15.
[8] R. K Ganti, F. Ye, and H. Lei. 2011. Mobile crowdsensing: current state and future challenges. *IEEE Communications Magazine* 49, 11 (2011).
[9] D. He, S. Chan, and M. Guizani. 2015. User privacy and data trustworthiness in mobile crowd sensing. *IEEE Wireless Communications* 22, 1 (2015), 28–34.
[10] K. Lun Huang, S. S Kanhere, and W. Hu. 2010. Are you contributing trustworthy data?: the case for a reputation system in participatory sensing. In *MSWiM*. 14–22.
[11] K. Lun Huang, S. S Kanhere, and W. Hu. 2012. A privacy-preserving reputation system for participatory sensing. In *LCN*. 10–18.
[12] H. Jin, L. Su, D. Chen, K. Nahrstedt, and J. Xu. 2015. Quality of information aware incentive mechanisms for mobile crowd sensing systems. In *MobiHoc*. 167–176.
[13] A. Josang and R. Ismail. 2002. The beta reputation system. In *Bled eConference*, Vol. 5. 2502–2511.
[14] J. Koo, R. Panta, S. Bagchi, and L. Montestruque. 2009. A tale of two synchronizing clocks. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*. ACM, 239–252.
[15] N. D Lane, Y. Chon, L. Zhou, Y. Zhang, F. Li, D. Kim, G. Ding, F. Zhao, and H. Cha. 2013. Piggyback CrowdSensing (PCS): energy efficient crowdsourcing of mobile sensor data by exploiting smartphone app opportunities. In *SenSys*. 1–14.
[16] Q. Li, G. Cao, and T. F La Porta. 2014. Efficient and privacy-aware data aggregation in mobile sensing. *TDSC* 11, 2 (2014), 115–129.
[17] C. Meng, W. Jiang, Y. Li, J. Gao, L. Su, H. Ding, and Y. Cheng. 2015. Truth discovery on crowd sensing of correlated entities. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*. ACM, 169–182.
[18] C. Meng, H. Xiao, L. Su, and Y. Cheng. 2016. Tackling the Redundancy and Sparsity in Crowd Sensing Applications.. In *SenSys*. 150–163.
[19] C. R Perez-Toro, R.K Panta, and S. Bagchi. 2010. RDAS: reputation-based resilient data aggregation in sensor network. In *SECON*. 1–9.
[20] Rajib Kumar Rana, Chun Tung Chou, Salil S Kanhere, Nirupama Bulusu, and Wen Hu. 2010. Ear-phone: an end-to-end participatory urban noise mapping system. In *IPSN*. 105–116.
[21] J. W Tukey. 1977. Exploratory data analysis. (1977).