CrowdS: Crowdsourcing with Smart Devices

V. Granfors, J. Waller, P. Mrazovic, and M. Matskin
Department of Software and Computer Systems
School of Electrical Engineering and Computer Science,
Royal Institute of Technology (KTH), Stockholm, Sweden

Abstract – In this paper we present design, implementation and experiments with a mobile crowdsensing system, called CrowdS, where smart devices (for example, smart phones or tablets) provide data requested by users of other smart devices. Actually each device can be a requester of data, a provider of data or both. Data requests can be passive (sensor tasks) and/or active (human intelligence tasks). The experiments with the platform were carried on a target group of users and system characteristics, as well as customer behavior, were investigated.

Keywords: smart mobile devices, crowdsourcing, mobile crowdsensing

Type of submission: Regular Research Paper

1 Introduction

The number of operating mobile personal devices is now counted in billions and their capabilities are growing in power. Currently it is assumed that they are used mainly for personal reasons: mostly communication, sometimes information gathering and organization. This means that at the moment they look like a collection of independent, opportunistic units, consuming data and information but not contributing to commonwealth or some collaborative goals. We are looking how to recruit devices to contribute to common goals and tasks. We assume that integrating such devices into the cloud computing environments may significantly extend service provision capability of the cloud environments. This is especially beneficial when the devices are carriers of different sensors and meters that provide opportunities for monitoring different environments. We provide a solution to this problem by applying a concept of crowdsourcing to a large collection of smart devices. This solution can also open opportunities for new types of cloud-based services. We assume that devices can involve humans to help them perform required task(s). Such a symbiosis of humans and devices opens possibility to utilize the human body as receptors and actuators of devices when people find that this is beneficial for them.

Let us imagine a case where continuous monitoring of some environmental characteristics (for example, the radiation or noise levels) is needed in an area where no sensory infrastructure has been installed. Naturally we would expect that an alarm be triggered if the level will exceed the level of natural background radiation/noise. Most cost-effective way to solve this task would be to outsource this to someone with measuring devices in this area (for example, this could be people carrying devices – smart phones or tablets). Availability of people with connected devices in the area is highly probable because people carrying mobile devices (and often private measuring devices) are nowadays available in almost every corner of the world and smart phone (with camera or microphone) could be used as low cost radiation/noise detectors. In this case we could outsource tasks directly to human’s devices. No direct involvement of device carriers will be required if their devices are networked. Thus instead of installing and maintaining a fixed (possibly expensive) sensory infrastructure for measuring the environmental conditions we can develop a flexible and dynamic networks of devices that are able to perform the tasks. In doing so we are going utilize device capabilities (that might be idle otherwise) on a regular base or in emergency cases. In the above case we considered usage of already existent sensors in the modern smart phones, however, with increasing number of sensors embedded into smart phones (or sensors connected to smart phones via Bluetooth) the number of measured environmental characteristics will increase. For example, crowdsourcing smart devices with pollen and dust sensors would allow provision of up to date information to people suffering with allergy by triggering alarms or delivering information about conditions in a particular area.

Other cases where our approach can successfully work include surveillance and emergency situation management in different geographical areas and optimizing of transport situation in a city involving citizens carrying smart devices.

It is important to underline that the devices (as well as humans carrying devices) may have their own agendas, missions and objectives, but in addition they may provide services that are not directly related to their main behavior. This reminds human societies where some tasks appear to be important while being outside the main activities of particular persons. In such situations motivated persons may work as volunteers for moral reasons or as short-time employees for “commercial” reasons. In our work we consider creation of motivation for smart devices emphasizing short-time employment reasons mainly while also taking into account a “volunteer” behavior model. In the context of this work we are focusing on smart mobile devices rather than on simple devices like just sensors. Smartness of devices assumes decision making ability, which
assumes existence of a processing unit, a memory unit and a communication unit. On the high end of our smart device spectrum we could assume autonomous mobile robots. However, on the low end of the spectrum we see smart phones or tablets that are smart enough for our purpose but not completely autonomous.

This work is a continuation of our previous work on general requirements for mobile crowd sensing environments [1] and task allocation methods for mobile crowdsourcing [2]. In this paper we underline details of the system implementation and experiments with the system.

2 Related work

2.1 Crowdsourcing of Smart Devices

Crowdsourcing was coined by Jeff Howe [3]. It refers to a way of sourcing a task to a group of potential agents via an open call instead of direct allocation of the task to a particular agent. The benefits of crowdsourcing are based on utilization of group intelligence. The key element of crowdsourcing is open call. Recently crowdsourcing became very popular in the context of the Web. One of the most successful examples of crowdsourcing application is Amazon's Mechanical Turk (AMT), the world's largest human computation platform. However, in the current research context crowdsourcing mainly assumes outsourcing of human resources while potentially the even more promising area of applying crowdsourcing for smart technical devices is mostly not considered. In connection to that we can mention the work by Yan et al. [4] who pointed out the enormous potential of crowdsourcing in the context of sensor-rich mobile devices, such as smart phones. The authors demonstrated a mobile crowdsourcing platform mCrowd that enables mobile users to post and work on sensor-related crowdsourcing tasks. The major shortcomings of this solution are first the need to hardcode support to specific sensors in the platform, second that human users are required to participate in the process, and third limited support for machine-oriented services. Research in this thread has led to a sample application, called CrowdSearch [5], which is an image search system for mobile phones. CrowdSearch combines automated image search with human validation of search results.

Several sources [6,7] refer to the concept of “passive crowdsourcing” which assumes that information can be gathered without asking humans to do something beyond their usual activity. This refers to collecting information about users carrying smart phones based on data available from the communication environment. Typical examples are users’ location (or traffic conditions) recognition from the available telecom or wireless networks. Another source is enabling users to send their GPS location to some applications [8,9,10]. The information about location can be quietly logged providing bases to analyses in such cases. While there is mentioning of a great potential of passive crowdsourcing, to the best of our knowledge, the proposed solutions look quite ad hoc. It is also recognized that passive crowdsourcing must put a serious attention to the privacy issues. Also the known cases don’t consider situation when users are involved in some common tasks and contribute to crowdsourcing for solving the task by demand.

Recently the areas of mobile crowdsourcing and participatory sensing have been gathering extensive attention in the research community. Related surveys can be found in [15] and [16]. Despite all the studies on crowdsourcing, only few works have focused on bringing the mobile crowdsourcing and participatory sensing to the real world.

2.2 Cloud computing infrastructures for sensors and smart devices

Cloud computing refers to a paradigm for resource provisioning on demand where resources may include software, infrastructure and/or platforms. Provision of these resources as services has benefits of scalability and performance, lower infrastructure and maintenance costs, storage capacity, universal access to resources, etc. Cloud computing is now considered as a promising way of service delivery and has a strong drive in the industry, including major software companies, such as Amazon, Google, Microsoft, IBM, SalesForce, SAP and others. Later [11] a notion of clouds of internet-connected objects was coined which refers to large-scale networks of spatially distributed entities with scalable processing and storage capabilities. Also there have been some efforts to integrate sensor networks to cloud data center recently. In particular, some results reported [12] where cloud infrastructure manages sensors via sensors virtualization. Another work [13] proposes publish/subscribe model for managing sensors networks in clouds for community-centered applications. There are also some works [14] on managing sensor data in cloud using annotation and metadata. At the same time there have been some industrial efforts in collecting, interpreting and sharing sensor data. For instance, CommonSense (http://www.sense-os.nl/crowd-sensing), developed by Sense Solution, is a cloud-based platform where users can connect any third-party sensor or sensing device to CommonSense. Although data belongs to the owner of the device, users can decide to share their data, either personally or anonymously. Feeds that can be used for sensing include smartphones, MyriaNed wireless sensor networks, any third-party sensor, GPS-trackers in cars and open data feeds. The above-mentioned approaches, however, mainly manage and give provision to static resources and services.

3 System Model

The proposed mobile crowdsourcing system is composed of a server, a set of smart devices which are used to make service requests, and a set of smart devices which are data providers. Requesters and providers do not have to be mutually exclusive, a device can be both a requester and a provider at the same time. The server side of the
system has the role of auctioneer in the task allocation, and is also responsible for quality control, reputation management, and reward management. There is no direct communication between smart devices in this system, all communication goes through the server.

![Diagram](image)

Figure 1. Mobile crowdsourcing system overview

The role of requester or provider of a smart device in the system is not predefined. All devices are considered a provider given they have the necessary requirements for a given task (such as desired sensors etc.). A user can whenever he/she wants create a task, and therefore be a requester.

3.1 Server side

3.1.1 Task allocation

The task allocation module is a coordinating component whose main responsibility is to identify and distribute tasks to a set of task-relevant information providers. The decision of what provider to choose is based on its location, quality and reputation. All this may greatly impact the quality of collected data. Task allocation may also play an important role in load-balancing and energy conserving. In the context of this paper, an auction based approach is used to allocate tasks to providers. CrowdS task allocation implements a sealed reversed auction with multiple winners which is common among many mobile crowd-sensing systems. The implementation is based on Vickery-Clarke-Groves (VCG) mechanism [18,19,20] and it is beneficial for truthful bidders because of in this case truthful bidding has the same or better utility than underbidding [2].

CrowdS handles task allocation as follows:

- A task requiring n number of providers is requested and sent to the core system.
- The core system finds all available and online providers.
- Filters out unsuitable providers, for instance:
  - Task giver
  - Missing necessary sensor
  - Too far away from the requested location
- If the remaining number of providers is less than the requested amount n, a failure notification is sent back to the task requester.
- Applies reversed VCG using the Task Allocation (TA) module:
  - If n equals the amount given to TA, simply return all of them. Their payment equals to their bid.
  - Otherwise
    - Sort providers by lowest bid
    - If two providers have the same bid, prioritize providers that have been online the longest.
    - Return the first n providers from the sorted list. Their payment equals to the bid of provider n + 1.

3.1.2 Reputation management

Crowdsourcing systems should strive to continuously recruit users to keep participation high. This, however, opens up a way for malicious users to make erroneous contributions. Therefore, to ensure high quality of data, the malicious behavior must be discouraged. Our approach to tackle this problem is to classify providers by their willingness to participate in the system and by previous experience with them. In ideal case we can assume that if a task is assigned to provider, then it should be completed. But that is not always the case, and some providers might be worse than others to complete their assigned tasks. This is why the system keeps track of the last 20 tasks distributed to each provider and calculates their reputation by dividing the number of completed tasks by the number of given tasks. Until the provider has been given 20 tasks, system keeps a base value of 0.5 for its reputation.

3.1.3 Quality control

The purpose of quality control is to ensure better quality of the contributed data. There are multiple ways to accomplish this depending on the nature of the data. Currently two types of data are considered by CrowdS: text and numeric. For textual data, the current implementation uses majority voting and for numeric values a distance formula is used.

In majority voting the majority decides the truth: if an answer option is chosen by the majority of the providers, then this option is considered as the truth. There are two types of tasks that currently return textual data: single and multiple choice tasks. Multiple choice tasks are split into decision tasks for each answer option. Each of those tasks has own inferred truth depending on whether it was chosen by majority or not.

Numeric values have to be handled differently from textual, since it is unrealistic to vote for a majority with unlimited number of possible choices. CrowdS uses the mean value of all the received values as the inferred truth instead, and calculates each provider’s quality by their distance from that value.

Each registered provider has a value representing overall quality of its contributed data, which is the average of its quality values calculated from the last 20 tasks.

3.1.4 Reward management

Participating in crowdsourcing activities drains the device’s resources, such as battery and computing power. Therefore, users participating in such activities need to be rewarded for their efforts and resource consumption. An incentive mechanism is also needed for keeping participation high. A common way to reward participation is to offer micropayments. The reversed VCG auction rewards the providers with the amount corresponding to the values of their bids or with the amount corresponding to the n+1 provider’s bid as payment if there are more than n providers to be chosen [2].
3.1.5 Requestor and provider

Users of the system act both as providers and requesters. These are two important roles with two different purposes. The requester creates a task, pays for completion and has access to the results. The provider replies to received tasks, either manually by providing input personally or automatically when gathering sensor data, and collects the promised payment afterwards.

3.1.6 Location

To find providers close to the target location, the distance from available providers is calculated using their last given location in longitude and latitude. Using this distance all providers within the radius of 25 meters from the basic search point are checked. If the number of providers found is at least as large as the number needed, all of the providers are returned (if there are more available providers than required then the providers that have been online the longest are prioritized). Otherwise the search area is increased by doubling the search radius, and providers in range are checked again. This process continues until either enough providers are found or the maximum default search radius of 800 meters is reached. If the later occurs, the search is stopped without task allocation.

3.1.7 Modularity

An important property that we tried to implement in CrowdS is modularity. This means that modules with a similar functionality could be easily substituted without having to reprogram the system. This is why a website was created to provide ability to configure the server. It makes possible adding a new module into a corresponding location and then choosing it as the current component instead the old one. Of course, to make this work all added modules have to implement a predefined interface with predefined functions that the core expects to be available. For instance, the current Task Allocation module which implements the function getUsers for finding information about the system’s users, can be replaced by another module depending on storage solution used.

3.2 Device side

The device side of the system is set up with a communications manager, a sensor task (ST) manager and a human intelligence task (HIT) manager as shown in Figure 2. The communication module handles all communications and direct all messages to the appropriate destination. The ST manager handles sensor tasks by collecting and sending desired sensor data and the HIT manager handles new and expired human intelligence tasks.

![Figure 2: Device overview (preliminary)](image)

The communication manager is responsible for the communication between server and smart device, and to deliver messages to appropriate modules. To achieve the communication between server and device, Google’s Firebase service is integrated into the system, in particular, the Firebase Cloud Messaging service (FCM). FCM provides a reliable connection between server and device that makes it possible to deliver and receive messages and notifications on iOS, Android, and the Web. With FCM it is possible to send notification messages that are displayed to a user. In the context of the system, notification messages are sent when a new task has been assigned to a device and when a created task has been completed. These notification messages are automatically handled and displayed by the FCM on the notification tray of the targeted device. Notifications have predefined set of key-value pairs that should be used.

The other types of messages that can be sent with FCM are called Data messages. Data messages are sent in cases when the server wants to communicate with a device. These cases are as follows: when there is a new sensor task, when there is a new human intelligence task, when there is an expired task, when sending heartbeat messages and when sending forced logout message (only used while testing). These messages have to be processed by the system, and have only custom key-value pairs. Notifications and data messages can be combined.

3.2.1 Registration

The registration process is simple. The person who wishes to register an account is asked to fill in his/her email address, a username and a password and answer three short questions. The questions asked during registration (see Figure 3) are used to determine the users bid during the task. Depending on the answer, the bid varies between 3 and 9. Because a reversed auction scheme is applied, the lower the bid the better chances are to perform a task. When everything is filled in and the 'register’ button has been pressed, the data is sent to the server.
3.2.2 Human intelligence task manager

The HIT manager is responsible for displaying assigned tasks, as well as sending the data from performed tasks and newly created tasks to the server. When a requester wants to create a task, he/she has to choose whether create a human intelligence task or a sensor task. If human intelligence task is chosen, the user is asked to choose between existing task types. Human intelligence task types implemented in the system include Single Choice, Multiple Choice and Numeric tasks. For Single choice tasks, user types a question to be answered and adds as many answer options as needed. In addition to the question and options, a location from which user wants to get answers is also required. This location is selected by clicking on the displayed map. The same sequence of actions is taken to create a Multiple Choice task. Numeric tasks do not require any options, as the intended answer to such task is a number. This means that only a question and a location need to be created.

It is quite intuitive to respond to human intelligence tasks assigned to a user. When a new task is received, it appears in a list that only contains assigned tasks which has not been answered yet. When selecting one of the tasks in the list, the question and available options appear on the screen. After giving an answer and pressing ‘Continue’ the answer is sent to the server where it is processed.

3.2.3 Sensor task manager

When creating a sensor task (ST), the user needs to perform the following steps: choose which sensor to use, choose how many readings to have, and choose a location which he/she is interested to gather data from (see Figure 4). When ‘Create task’ is pressed the information is sent to the server. The ST manager takes care of incoming sensor tasks by gathering sensor data and sending it to the server.

In contrast to assigned human intelligence tasks, sensor tasks are performed automatically. When a new sensor task is received by the communication manager, it is passed along to the ST manager who identifies which sensor has to be used. When the requested amount of readings has been performed, the data gathered from the identified sensor is sent to the server. Because Apple does not provide any API for environmental sensors, such as light or pressure, this feature has not been implemented for the iOS version of the system yet.

3.2.4 Task history and ongoing tasks

Besides functions for creating tasks and performing assigned tasks, there are functions for viewing a task history and ongoing tasks. The task history function shows all completed tasks that the user has created. If a task in the list is selected, the question and the answer are shown on the screen along with the time of creation and completion. The ongoing task function shows all tasks the user has created which are not completed yet.

3.2.5 Modularity

In the case of mobile devices, we have achieved modularity by making it easy to add, remove or change available sensors for sensor tasks and task types for human intelligence tasks. When adding a new type of human intelligence task, it is required to provide a module which defines its interface with a user. When adding a new sensor type for sensor task, it is required to add it to the list containing all existing sensor types, together with its numeric value (e.g. 1 for accelerometer).

4 Results and Discussion

4.1.1 Experiment

The first larger test for CrowdS was executed during the end of the year 2017. Its main purpose was to gather information about how the system performs regarding fairness and coverage. This in particular includes answering to the following questions: How fairly were providers chosen for a task? How large should be a search area in order to find a provider? The participants of the experiment were students from a university course and some other volunteers. Every participant was asked to create 50 human intelligence tasks (HIT) and 50 sensor tasks (ST), and wait for their completion. Tasks that expired or terminated...
without an answer are not considered to be completed and they had to be redone.

The following list contains the parameters that were used for the test:

1) Radius of the search area:
   a. Initial value: 25m;
   b. Maximal value: 800m;
   c. Increment: doubling

2) Time to complete:
   a. HIT: 3h
   b. ST: \((n-1) \times 9 + 15\) minutes, \(n=1,2,..\)

3) Providers:
   a. Exactly one
   b. Shown on map

4.1.2 Participation

The number of participants in the test was 21, where 16 used an Android device and the remaining 5 used a device running iOS. Out of these 21 people, only 5 completed the whole assignment.

4.1.3 Task distribution

There was almost an even distribution between the amount of Human Intelligence Tasks (HIT) and Sensor Tasks (ST) created. This is expected since the participants were asked to create 50 of each.

More than half of all created HITs were numeric tasks. The simple reason behind this majority (62.7) most likely was because these tasks were faster to create. Due to the nature of numeric tasks they only require a question to be provided while single and multiple choice tasks need provision of answer options as well. In conjunction with taking the path of least resistance this was probably an explanation why numeric tasks were most popular among the participants.

The two types of Sensor Tasks: Light and Pressure, had almost the same amount of tasks completed. Both require an equal amount of effort to create and should therefore not affect how many of each type were created, the decision was based on the users’ preferences instead.

4.1.4 Coverage

Figure 5 shows how big the search radius was when the required number of providers was found to create a task. In the ideal case it would be nice to have as small radius as possible while always finding enough providers to allocate the task. This was not the case and most of the time the search area had to be extended to the maximum of 800 meters before finding enough providers. The information about how many times the marker on the displayed map had to be moved due to not finding enough providers was not recorded. At the same time the location of all nearby providers was shown on the map with markers during selecting a location for a task. The reason why we decided to show everyone nearby on the map was to make it easier to create tasks. Otherwise with so few providers online during the test it would be too difficult to find anyone in desired location. We believe there are two main reasons why most of the tasks were created with a search radius larger than initial. The first one is that the participants did not prioritise precision while choosing a location for a task, as long as the task was created the location mattered little. The second reason was that the map did not let markers be placed on top of each other, which led to placing the markers not exactly on the targets.

4.1.5 Fairness

The fairness of the participants’ earnings was also measured. Since we had no knowledge of how much the resources spent are worth for the provider, it was hard to know the exact value of their earnings. However, we knew for how much they are willing to work - their bid. Assuming that the bid is greater than the resources spent, their earnings are included in the bid as well. So we normalize the earnings by dividing the cost of “hiring” the providers with their bid. Jain’s fairness index [17] for the earnings was 0.941. 61 out of 802 tasks used the task allocation scheme to decide the price of the task. The rest had the same cost as the selected users bid. This in conjunction with a very low variance in user bids results in a very fair result.

4.1.6 Prices

The fairness of the prices of tasks was measured as 0.944 on the Jain’s fairness index. The price of the task equals the sum of the chosen provider’s bids. Like the reasoning about earnings, 741 out of 802 tasks had the users bid equal to its price. Because the variance in bids is low the fairness of prices is high.

4.1.7 Response time

This is how long it took for a service provider to respond on a HIT. The vast majority of tasks were completed within 10 minutes from creation. The median is 212 seconds (3.53 minutes), which shows that the participants in the test were very fast in responding to assigned tasks. No tasks were completed after 3 hours as they were expired.

Figure 5. Search radius depending on the number of tasks
5 Conclusions and Future Work
This paper presents a design and implementation of a
crowdsensing system CrowdS which allows create both
sensing and human intelligence task and allocate them to
smart devices (and their owners) in a community network. In
the system each device (user) can be requestor of data,
provider of data or both. The system supports both iOS and
Android enabled devices.
The experiment that we carried out demonstrated a
high acceptability of the system by the users. The advantage
of the CrowdS system, in comparison to already available
network services of similar type, is in a combination of
sensing (automatic) tasks with human intelligence tasks
(which require human actions). Such combination provides
powerful opportunities for solving of real world problems.
For future work we are going to develop a more
sophisticated reputation management mechanism, fine-tune
our reward mechanism and provide a more optimal way of
selecting the providers.

6 References
Combining Crowdsourcing and Participatory Sensing. 2015
IEEE 39th Annual International Computers, Software &
Applications Conference (COMPSAC2015), IEEE
Computer Society, Taichung, Taiwan, July 1-5
Mechanism for Mobile Crowdsensing with Smart Consumer
Devices". In Proc. 4th IEEE International Workshop on
Consumer Devices and Systems (CDS 2016), COMPSAC
2016 Workshops, Atlanta, USA, 2016
Is Driving the Future of Business Three Rivers Press. New
York. 2008
Corner. 2009. mCrowd: a platform for mobile
crowdsourcing. In Proceedings of the 7th ACM Conference
on Embedded Networked Sensor Systems (SenSys '09).
CrowdSearch: exploiting crowds for accurate real-time
image search on mobile phones. In Proceedings of the 8th
international conference on Mobile systems, applications,
and services (MobiSys '10). ACM, New York, NY, USA,
77-90.
crowdsourcing.html (accessed: 2017-12-15)
subway-arrival/ (accessed: 2018-02-15)
03-20)
[9] K. Arnold, B. Osullivan, R. W. Scheifler, J. Waldo, and
Addison-Wesley, Reading, MA, June 1999
[10] Salutation Architecture Specification (Part 1), V2.0c
https://www mỹumpu.com/en/document/view/41060143/saluta-
tion-architecture-specification-computer-systems-colorado
(accessed:2018-03-20)
(accessed:2018-03-20)
-Physical Sensor Management with Virtualized Sensors on
Cloud Computing in: Network-Based Information Systems
(NBiS), 2010 13th International Conference on Issue Date:
14-16 Sept. 2010, pp: 1 – 8
framework of sensor-cloud integration opportunities and
challenges, ICUIMC '09 Proceedings of the 3rd International
Conference on Ubiquitous Information Management and
Federated Sensor Networks and on building front-end
interface to data clouds 2010 IEEE International Conference
on Sensor Networks, Ubiquitous, and Trustworthy
Computing (SUTC). PP. 107-114
Privacy, security, risk and trust ,2011 IEEE third
international conference on social computing (socialcom).
[16] N. D. Lane, E. Miluzzo, H. Lu, D. Peebles, T.
Choudhury, and A. T. Campbell, “A survey of mobile phone
sensing,” Communications Magazine, IEEE, vol. 48, no. 9,
Measure of Fairness and Discrimination for Resource
Allocation in Shared Computer Systems”, Digital Equipment
[18] W. Vickrey, “Counterspeculation, Auctions, and
8–37, Mar. 1961.