Aspects and Challenges of Mobile Crowdcomputing

CARRON, Blaise, LEONE, Pierre, EVANGELATOS, Napoleon-Orestis

Abstract

In this paper, I discuss what is crowdsensing and crowdsourcing and how they work from a high level perspective. I show examples of applied applications and detail some of them. Having such power and abilities is a great thing but it comes with a price and therefore I also discuss the challenges that mobile sensing applications have to face. Finally, I have a glimpse at the near future brought by this revolutionary field.

Reference


Available at:
http://archive-ouverte.unige.ch/unige:75924

Disclaimer: layout of this document may differ from the published version.
Aspects and Challenges of Mobile Crowdcomputing

Université de Genève

Author: Blaise Carron
Professor: Pierre Leone
Teaching Assistant: Orestis Evangelatos

January 21, 2015
# Contents

1 **Introduction** 4

2 **Crowdsensing** 4
   2.1 Definition ............................................. 4
   2.2 Characteristics ........................................ 5
   2.3 Sensors .................................................. 6
      2.3.1 Inertial sensors .................................... 6
      2.3.2 Positional and user proximity sensors .......... 6
      2.3.3 Ambient sensors ................................... 6
      2.3.4 Other sensors ..................................... 7
   2.4 Applications ............................................ 7
      2.4.1 Environmental ...................................... 8
      2.4.2 Transportation ..................................... 8
      2.4.3 Infrastructure ..................................... 8
      2.4.4 Social and health .................................. 8
      2.4.5 Applications’ architecture ....................... 9
      2.4.6 Application scale .................................. 9
   2.5 Challenges .............................................. 10
      2.5.1 Technical limitations and diversity among devices 10
      2.5.2 Privacy ............................................. 11
      2.5.3 Incentives .......................................... 11
      2.5.4 Solutions .......................................... 12

3 **Explanation of the process / Architecture** 13
   3.1 Sensing the data ....................................... 13
      3.1.1 Participatory sensing ............................... 14
      3.1.2 Opportunistic sensing ............................. 14
   3.2 Preprocessing (local analytics) ....................... 14
      3.2.1 Calibration ......................................... 14
      3.2.2 Feature extraction .................................. 14
   3.3 Privacy .................................................. 14
   3.4 Aggregate analytics .................................... 15
      3.4.1 Classification (Context inference) ............... 15
   3.5 Client applications ..................................... 15

4 **An example: Improving Wireless Network Performance Using Sensor Hints** 16
   4.1 Architecture ............................................ 16
4.2 Hint-aware bit rate adaptation ........................................ 17
4.3 Hint-aware AP association ........................................... 18
4.4 Topology maintenance ................................................. 18
4.5 Vehicular network path selection .................................... 18
4.6 Limitations ............................................................. 19

5 Crowdsourcing .......................................................... 19
5.1 Definition and characteristics ........................................ 19
5.2 Execution of the process ............................................. 20
  5.2.1 Registration and task specification ......................... 20
  5.2.2 Task distribution ................................................. 20
  5.2.3 Task resolution .................................................. 20
  5.2.4 Task evaluation .................................................. 21
5.3 Examples ............................................................. 21
5.4 Challenges ........................................................... 22

6 Conclusion ............................................................... 22
1 Introduction

Today, mobile devices are everywhere. Millions of people carry them all day long in their everyday life, at work, during their free-time, when they are travelling and even when they stay home. In 2013, more than one billion smartphones were shipped [5] and most of the world is covered by these devices, creating a big network of a new kind. The computational power and the mass availability of modern mobile devices permit a whole lot of new applications to be discovered and made.

Since people are closely related to those devices, it enables a new level of applications using human intelligence to achieve higher level task such as seen in crowdsourcing. All these devices are embedded with cheap sensors that can give very useful informations about the device’s surroundings and activity thus enabling more complex and innovative applications to be developed.

In this paper, I discuss what is crowdsensing and crowdsourcing and how they work from a high level perspective. I show examples of applied applications and detail some of them. Having such power and abilities is a great thing but it comes with a price and therefore I also discuss the challenges that mobile sensing applications have to face. Finally, I have a glimpse at the near future brought by this revolutionary field.

2 Crowdsensing

2.1 Definition

The definition below, which sums well the ambition and the possibilities of mobile crowdsensing, is given by the IEEE Communications Society (ComSoc), a reference in communication technologies, for their related “Call for Papers”. The ComSoc has published many scientific papers in magazines and held numerous events since 1952 to promote the advancement of all communication technologies [1].

Mobile Crowd Sensing (MCS) presents a new sensing paradigm based on the power of various mobile devices/objects (e.g., smartphones, wearable devices, sensor-equipped vehicles, etc.). The sheer number of mobile device users and their inherent mobility enables a new and fast-growing sensing paradigm: the ability to acquire local knowledge through sensor-enhanced mobile devices - e.g., location, personal and surrounding context, noise level, traffic conditions, and in the future more specialized
information such as pollution - and the possibility to share this knowledge within the social sphere, practitioners, health care providers, and utility providers such as municipalities for example. The information collected by the mobile devices on the ground combined with the support of the cloud where data fusion takes place, make mobile sensing a versatile platform that can often replace static sensing infrastructures, and enabling a broad range of applications from urban dynamic mining, public safety, environment monitoring, just to name a few. [4]

Three important points come out of this definition and need to be taken into consideration. Firstly, the amount of devices used in crowdsensing is much higher than ever before. Secondly, thanks to the emergence of the cloud and social networks, the data produced is easily shared between this broad number of devices. And finally, crowdsensing paves the way to many new applications.

2.2 Characteristics

Given its mobile nature, crowdsensing has specific characteristics that can be used to explore much more complex applications than usual mote-class sensors[1]. Current mobile devices such as smartphones have much more computing power than mote-class sensors [15]. In the recent years, the number of mobile phones has exploded and nowadays almost everyone carries such a device, forming a network of millions of devices. In consequence, large scale sensing application deployment requires less effort and is much cheaper. Since humans are directly involved in mobile crowdsensing, more complex and versatile data can be collected [14].

However, those advantages are balanced by some shortcomings. Unlike classic sensor devices, mobile devices are dynamically moving and their availability is not guaranteed. People can also decide to turn off their phones, disable data sharing or travel with them. Therefore, an area that is usually covered may no longer be. The quality of the data produced by mobile sensors in terms of accuracy, latency and confidence varies from one device to another and is not stable in duration. Each manufacturer decides what type of sensors they embed in their devices and with what precision, creating a very disparate network of sensors.

1A sensor node, also known as a mote (chiefly in North America), is a node in a wireless sensor network that is capable of performing some processing, gathering sensory information and communicating with other connected nodes in the network. A mote is a node but a node is not always a mote. [6]
Other important concerns like privacy and resource limitations are discussed in the challenges section of this paper.

2.3 Sensors

In this section, I describe the most common sensors found in the majority of modern smartphones. As explained in [14], sensors can be classified in three main categories:

2.3.1 Inertial sensors

In this category, two types of inertial sensors can be distinguished: accelerometer and gyroscope. Accelerometers can provide precious information about the physical state of a user (e.g. walking, running, standing, sitting, etc.) by analyzing the frequencies and the direction of the accelerations the mobile is under. Gyroscopes can be used to determine the direction and the body position and are useful for example in stabilizing the camera of the mobile.

2.3.2 Positional and user proximity sensors

Positional and user proximity sensors concern mainly bluetooth, WiFi, cellular signal and GPS. Bluetooth is primarily used for short range communications, device detection and recognition, by obtaining the Bluetooth MAC address, the device type and the device name from surrounding devices. WiFi however provides longer range communications than Bluetooth and is faster. Access points MAC addresses can be used to determine the device location even inside a building. As regards to cellular signal, each cell tower has a unique ID that can be used to determine an approximate user location. By logging the different towers from which the cell signal is acquired one can determine the user’s trip. The signal strength pattern can be used to determine the user’s activity. A more precise location of the device can be obtained with the GPS. Unfortunately, the latter is power-hungry and does not work very well indoor.

2.3.3 Ambient sensors

The four ambient sensors taken into consideration in this section are: camera, microphone, light sensor and magnetometer. The camera can be used in recognition and identification applications. However, since image processing needs high computational resources and has high energy consumption, these applications often need a backend server to handle the analytics. In addition, the microphone is also very useful to gather information about the device’s surroundings (at home, at a coffee shop, etc.). Moreover, with the use of the microphone we can log the noise pollution
of an area or assess the user’s social activity. The light sensor is useful to determine ambient luminosity and therefore to infer the device’s context (laying on the desk, in the user’s pocket, etc.). The magnetometer is used to detect the earth’s electromagnetic field and therefore can be used to determine an object’s orientation or to perform indoor navigation.

2.3.4 Other sensors

The sensors described earlier are the ones that are most commonly found in smartphones but there are many others that are used in specific device such as the Mobile Sensing Platform (MSP). As Choundhury and al. present in [10], “[t]he MSP is a small wearable device designed for embedded activity recognition with the aim of broadly supporting context-aware ubiquitous computing applications”. In addition to a microphone, a light sensor, an accelerometer and a compass, this device includes a barometer, a thermometer, a humidity and an infrared sensor.

Other sensors already used in MCS applications include air pollution, blood pressure and neural activity. These sensors are not yet embedded in the current market’s common smartphones but as their costs and size decrease, they will progressively make their way to new mobile devices. [21]

2.4 Applications

There are a lot of applications that can be developed using a broad variety of sensors available in modern mobile devices. Firstly, I present a few examples of applications before elaborating on their different types in the domain of MCS and their potential. For example, by gathering information from a crowd of devices, an application could monitor the air and noise pollution, build real-time traffic patterns, detect pot holes on the roads, identify road closures and calculate transit timings [15]. As described in [23], by using movement hints to enhance current WiFi protocols, it is possible to increase the connection’s throughput and make it more stable while moving. A company could use the light and positional sensors of its employee’s mobile phones to reduce energy consumption in its offices by turning off the lights or maintaining at threshold levels the air conditioning when all employees leave an office.

Crowdsensing applications can therefore be classified in multiple categories according to [21] [15]:

7
### 2.4.1 Environmental

The first category is environmental applications such as air or noise pollution monitoring. Other examples include carbon emission monitoring, controlling of water levels and observing wildlife habitats. One demonstrative example with broad results is GasMobile [17], a project conducted by ETHZ to create collective air pollution maps. Since smartphones do not include gas sensor, they have built a USB ozone sensor made with low cost hardware. The results from their experiments showed that precise air pollution monitoring is feasible with low cost hardware.

### 2.4.2 Transportation

The second category is transportation applications such as traffic congestion detection, dynamic route planning and accurate travel timing. A real world example is the VTrack project [25] carried by the MIT. In this project, smartphones are used to detect traffic hotspots (portion of roads where the travel time is much higher than expected) and to share them in real time with the VTrack server. Then, a user can use the application to determine the travel-time and choose the route with the minimum time value.

### 2.4.3 Infrastructure

The third category concerns infrastructure applications, for example road conditions monitoring (honking level, potholes), parking availability checking and outages of public works detection (fire hydrants, traffic lights). Parknet [22] is an example of an infrastructure application, which provides a spot-accurate map of parking availability. To build the map, Parknet uses vehicles equipped with a GPS and an ultrasonic sensor on the passenger’s side to detect empty spots along the road. The data is then uploaded and processed by the Parknet server and clients can query the system to find a parking spot.

### 2.4.4 Social and health

Finally, the last category regroups social and health applications. Some representative examples are physical exercise monitoring, weight logging and eating habits management (useful for people suffering from diseases like diabetes). These applications often share data with social networks (or use their own social platforms) to motivate their users and encourage them to change their habits. Sensors can also be used to detect, classify and share life events on social platforms such as Facebook or Twitter. UbiFit Garden [11] is an example of this category. It is a joint project conducted by Intel and the University of Washington to encourage physical activity.
The system is based on the MSP, to detect physical activities in real time. Data is then sent to a smartphone to be processed via Bluetooth. The system displays as background on the mobile phone a garden’s picture and rewards the user by adding to it flowers and butterflies when the latter logs activities and achieves goals.

2.4.5 Applications’ architecture

As mentioned above, there is a large variety of applications which share many challenges in data collection, resource allocation and energy conservation. Unfortunately, as discussed in [15], there is currently no common architecture among the mobile crowdsensing applications. This situation delays the development of new applications and results in an increase overhead due to multiple heterogeneous applications trying to collect data from the same sensors instead of sharing their data samples. To avoid over sampling and writing different versions of local analytics and to provide the same API for every OS, the authors of [15] propose a new architecture. In order to speed up the development of new applications, this architecture should provide a high level language to specify data needs.

This architecture coordinates all the applications’ needs to optimize power consumption and resource allocation by sharing data samples. It can identify the set of devices that provide the desired data and send instructions to the devices accordingly. It can also adapt to changes in the devices’ network and accommodate the set to ensure the desired quality. This type of architecture is further developed in [27].

2.4.6 Application scale

Mobile sensing applications can operate at different contexts, from personal use to big communities [21]. At a single user level, applications are often used to collect data such as activity logging. When multiple users share common interests, they form a group. Group users share data within the group in order to achieve a common goal such as neighbourhood safety. To assess large scale phenomenon like traffic congestion in a city, an important number of users is needed. Since there are a lot of people involved in them, community applications need to be scalable and as the scale grows so does privacy concerns, due to the need of sharing personal data with more and more people.
2.5 Challenges

As we have seen before, the scope of applications provided by crowdsensing is very broad. However, there are some challenges and obstacles that need to be faced.

2.5.1 Technical limitations and diversity among devices

One of the biggest technical issue is energy consumption. In fact, mobile devices have limited battery life and some sensors, like the GPS or the Camera, can greatly reduce it. Hence, there is a need for finer and more efficient control of sensors, for instance to manage sampling rate, reduce the power consumption when using the GPS, etc. As the demand grows, these features will eventually come [21].

Since crowdsensing is still in its infancy, the diversity between devices is considerable. Each device has its own subset of sensors and the control over them provided to the developer depends not only on the operating system, but also on the manufacturer. While considering these diversities, the developer of a smartphone based crowdsensing application has to implement different functions for connecting different sensors to the same application and also calibrate them properly. Therefore, there is a need for cross platform APIs and a common architecture as discussed earlier [23].

While the batteries’ capacity and CPUs’ performance are slowly increasing with time, smartphones are still limited in resources. Some algorithms can be computationally intensive and need power-hungry sensing rate. Then, the transfer of the acquired data consumes large amounts of power. Therefore, crowdsensing applications need to be energy efficient, minimize the CPU load induced by probing the sensors and manage the processing of the data to avoid conflicts with other applications.

In certain circumstances, Smartphones’s physical location can be very unpredictable, for instance, it can be in a users’ pocket or in their bag, the user can be travelling very fast, etc. Thus, the data produced by a MCS application can be unusable or burdensome to process. Facing that challenge, data mining algorithms are efficient in low noise and predictable environment but have trouble adapting to the dynamic conditions found in mobile sensing environment [21].
2.5.2 Privacy

A big non technical challenge in MCS applications is privacy. By collecting personal data from devices that are always carried by people, it becomes easier for a person or another application to track a person’s behaviour [23]. The positional hints used in many crowdsensing applications can reveal the home or work locations of users and the routes they often take, putting them at risk [15]. Moreover, other sensitive private informations can be found by analysing data from all the sensors, especially from the microphone and the camera. It is therefore very important that every MCS application assesses and resolves the different privacy concerns that the sensed data generates.

2.5.3 Incentives

An important factor for crowdsensing applications is to give incentives for the users to employ the application and share their sensed data. This is especially the case in participatory sensing which requires a greater participation of the users. Incentives, like micro-payments and other monetary rewards as coupons, perform well but are not easily scalable to large community applications. Social media, such as Facebook or Twitter, are good leverage for user’s participation and are useful to share information and build competition amongst fellow users. [21]

A credit system can be made, like in APISENSE [16], where users get credits when they upload datasets depending on the number and the nature of the sensors used. Users are then given badges and ranks to award them and could even earn rewards like coupons. Of course, the main incentive for applications offering a useful service comes by the benefit gained by using it. For instance, shortening users’s travelling time to a specific location by using the data shared by them and all the other users. WAZE [3], for example, combines user’s interest (shorter travel time) and a points system which rewards users when they report traffic hazards. Users are then ranked and can compare their score with their friends in a scoreboard, which stimulates competition amongst users and thus increase their usage time.

In [8], a group of researchers identifies the three underlying components of a MCS application, namely the task, the server and the crowd, and defines functions that regulate their interactions. The crowd is composed of a set of devices embedded with various sensors. The task represents the work that the server is wanting to execute using the devices in the crowd and can be divided in smaller equally sized segments. The server represents a stakeholder and tries to manage efficiently the budget at its disposal by attributing the optimal incentive for each task segment.
Each agent composing the crowd can freely decide to execute a task segment or not. The searchers have defined multiple utility functions defining the utility of a task segment considering different scenarios. They have also defined multiple incentives policies trying to give the optimal incentives given a task segment, a cost function computing the cost of a task segment for a user and join policies defining if a user will execute the task segment. After conducting experiments in Matlab, they have showed that some combinations of incentive policy and utility functions are more efficient to manage the server’s budget and achieve higher quality while giving the crowd adequate incentives. These results are significant when defining a MCS application and show key aspects of determining incentives.

2.5.4 Solutions

Scientists, developers and researchers have already started to dig into solutions for most of the challenges on the MCS applications. Offload processing calculations in a backend server can reduce CPU load. To improve battery lifetime, a lower sampling rate can be found by balancing accuracy and latency. Applications adapting to the context can also help to manage the battery by waiting to be in charging mode to process or upload the data. In order to save power and bandwidth, context problems, where a device produces very noisy data, need to be fixed. It is possible for instance to use nearby devices to control the data sensed by a single device, or use sampling algorithms which control the quality of the acquired data, to remove samples below a fixed threshold [21].

Different approaches are proposed in [15] to provide privacy for the users. The first is anonymization in which the application removes data identifying the users before sending them to the server. However, this approach is not very useful in many MCS applications because the users’ details are needed. The second is cryptography in which, by encrypting data before sending them to the server, the user’s privacy is preserved. However, encryption algorithms are CPU intensive and require the maintenance of key pairs which is not scalable. The last one is perturbation. The idea behind it is to add noise to the data in order to preserve the user’s privacy while not affecting the statistical computations needed for the application.

In [20], Krause and al. have characterized an approximation to determine near-optimal sensing policies taking into account the users’ preferences concerning privacy and power usage, the utility of the sensed data and the availability of devices. To determine a set of optimal observations, they have developed theoretical models to define phenomenon’s prediction, sensors’ availability and applications’ demand.
They have applied their work on a traffic monitoring application using data of Seattle’s surrounding highways’ speeds from 2006 and 2007. Experimental results show that it is possible to reduce significantly the amount of sensed data needed to achieve a certain level of accuracy, by optimizing the selection of observations.

3 Explanation of the process / Architecture

Most of the mobile sensing applications depicted earlier follow the same basic scheme that I describe in this section. The figure illustrates the different steps involved in this process and shows the high level architecture of a crowdsensing application. Each element in the figure is described in the following subsections.

![Figure 1: Mobile Crowdsensing Architecture](image)

3.1 Sensing the data

The process starts with a crowd of persons using a given application on their mobile devices. First, the application starts monitoring the desired sensors and collecting data from them. It is important to identify two different categories for sensing: participatory sensing and opportunistic sensing [14].
3.1.1 Participatory sensing

In participatory sensing, the users are directly involved in the process of obtaining the data. Thanks to this, the calibration and classification processes can be greatly reduced. However, participatory sensing depends on the disposition of the users to perform the sensing and therefore requires incentives to motivate them. Furthermore, data is biased by the users’ technological background and preferences.

3.1.2 Opportunistic sensing

In opportunistic sensing, the data does not require the users’ involvement to be acquired and therefore the data produced is generally more reliable [14]. The burden is easier to accept for the users since the data collection is effortless. However, the sensing has to be transparent to the users in order not to affect them.

3.2 Preprocessing (local analytics)

The raw data coming from the sensors is often noisy and jumpy. Hence, it needs to be filtered and converted in order to be used. This is done by preprocessing the data and has two main benefits: less energy and bandwidth is needed to transfer the data to the backend, and it requires less postprocessing [14].

3.2.1 Calibration

Unlike traditional sensors that are fixed, mobile devices tend to be used in many different environments and the data obtained is sensitive to their positions and orientations. For example, the accelerometer will not perform the same if the smartphone is in the user’s trousers or chest pocket. It is then required to have a detection stage before using the data or to train classification algorithms for all possible situations.

3.2.2 Feature extraction

The raw data is converted in a more computer friendly form called ”features”. The features are used by classification algorithms to infer the context. In addition, transferring features typically uses less bandwidth than raw data [14].

3.3 Privacy

Before sending the data to the backend, the application must protect its users. As discussed before, there are multiple methods to achieve privacy depending on the application’s goal and scale.
3.4 Aggregate analytics

Then, in order to achieve the desired goal, for example to monitor traffic in the centre of a city, analytics must be run at the backend to aggregate data from all the mobile devices. Data mining algorithms are involved in the process of handling large amount of data and those may require high storage capacity to store all the data collected by the mobile devices or require stream data mining algorithms [15].

3.4.1 Classification (Context inference)

Features derived from sensor data are then inserted into a classification algorithm (supervised or unsupervised). Classification algorithms are categorized in two types: discriminative (Discriminant analysis, Bayesian networks, Hidden Markov Model) or generative (Neural networks, Decision trees, hierarchical thresholds, Fuzzy logic, Clustering) [14].

Supervised learning classification uses labelled examples to train the algorithm and then fits the features extracted from the sensors to infer a class. For example, a class can be the current state of a user, e. g. walking, running, sitting, etc. These algorithms are useful because they can clearly define the different classes used by the model. A main disadvantage is that they do not scale well, which is a key requirement for large scale crowdsensing applications that involve a broad range of different people (age, physic, etc.). Other learning algorithms, such as semi-supervised or unsupervised, are more scalable but can generate classes that are not relevant to the application [21]. One way to accommodate this issue is to ask the users to label themselves the different classes and therefore the algorithm can scale well in an heterogeneous population. However, this increases the computational needs.

As for many other aspects of crowdsensing, there is a lack of a structure for data mining algorithms, different groups of searchers are developing and hand tuning their own classifiers and this results in a big waste of time and resources. There is a need for a global platform to share algorithms and data sets for common problems such as recognizing human activities (walking, running, sitting, etc.). [21]

3.5 Client applications

Once the data is processed, the client software can query the server to obtain the desired information, for instance the quickest route to a destination or a map showing air pollution.
4 An example: Improving Wireless Network Performance Using Sensor Hints

One very interesting applied example in the context of mobile sensing is the work done by the MIT Computer Science and Artificial Intelligence Laboratory described in [23]. They have used what they called sensors hints from an android mobile phone to improve WiFi performances. A sensor hint is a simple piece of information concerning the device’s movement. The different types of hints are described below. To achieve higher throughput and better stability they have developed new protocols using these hints with significant results. Although this example does not involve a big community of users, it demonstrates many key characteristics of MCS applications.

Mobile devices, such as smartphones, often switch between moving and stationary states, thus causing wireless networks to adapt to this behaviour. The device’s motion quickly modifies the wireless connection’s parameters; the channel quality changes rapidly, packet losses are much more frequent and the network topology is frequently modified. To overcome those issues, nodes should not keep long histories and routing tables should adapt promptly to changes in the network [24]. But when the device returns to a stationary state, these strategies will not be optimal. By creating a sensor-augmented wireless architecture, which provides protocols with movement informations about the device and the surrounding nodes, the network can adapt accordingly to the device’s current state and improve the wireless performances.

4.1 Architecture

As shown in figure 2, the architecture is based on the wireless stack but augments all its layers with hints from external sensors. The augmented protocols are using hints sent by the Sensor Hint Service (SHS) that abstracts the details of retrieving those hints for easy protocol development on different platforms. Two nodes can communicate together via the Hint Transport Layer of the SHS and protocols of the wireless stack can register to the SHS in order to receive or send hints to other nodes.

As described in figure 3, there are five hints used by the augmented network. The first one is movement and it defines if a device in moving or not. It is used for bit rate adaptation and topology maintenance. The second is walking and it detects if a user is walking or if he is using another mean of transportation. It is useful for AP
selection. The third hint is heading and is also used for AP selection. It measures the direction to where the user is moving and can also be used for vehicular path selection. The fourth one is speed and is used for path selection. The last one is environment and it determines whether the user is indoor or outdoor and is used for AP association.

<table>
<thead>
<tr>
<th>Hint Type</th>
<th>Hint Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Movement</td>
<td>True/False</td>
</tr>
<tr>
<td>Walking</td>
<td>True/False</td>
</tr>
<tr>
<td>Heading</td>
<td>Degrees Relative to True North</td>
</tr>
<tr>
<td>Speed</td>
<td>Miles per Hour</td>
</tr>
<tr>
<td>Environment</td>
<td>Indoor/Outdoor</td>
</tr>
</tbody>
</table>

Figure 3: Hint types exposed by the Sensor Library. Figure reproduced from [23], page 2.

4.2 Hint-aware bit rate adaptation

When moving, a device needs to adapt its bit rate quickly to suit the dynamically changing wireless conditions. RapidSample is an algorithm that reduces the bit rate after the first packet loss and quickly attempts higher bit rate after a few successes at
the current rate. While this strategy is not new, the Hint-aware bit rate adaptation switches between RapidSample for the moving case and SampleRate [9] for the static case using the movement hint. In mixed static/moving tests performed in [23], the hint-aware protocol performed significantly better even when using a simple binary movement hint.

4.3 Hint-aware AP association

Usually, a wireless client will scan one time all the available access points to find which one of them has the strongest RSSI (Received Signal Strength Indicator) value and then associates with it. If the RSSI value falls under a threshold, the client will start the process over. While this scheme performs well in a stationary case, it lacks when the device is moving. To maximize throughput, the hint-aware scheme uses three improvements over the standard protocol: scanning for a better AP every time a client stops moving, scanning more often when moving and never scanning when the device is in a stationary state. These techniques result in a 30% increased throughput. Moreover, to minimize handoffs, the hint-aware scheme uses a heading hint to select the one AP towards which the client is moving, resulting in 40% less handoffs and 10% throughput improvement.

4.4 Topology maintenance

To maintain the topology on wireless networks, each node sends probe packets to build a list of neighbours and the connection quality to them. To avoid wasting bandwidth by sending too many probes, a balanced approach needs to be found between topology accuracy and bandwidth usage. The hint-aware protocol adapts the rate of probe packets using a movement hint to maintain an accurate delivery probability even when moving. Having a wrong delivery probability can heavily reduce the throughput. In the conducted tests, the standard protocol estimation was wrong more than 30% of the time, whereas the hint-aware protocol stayed within 5%.

4.5 Vehicular network path selection

To avoid overhead and latency in vehicular networks caused by broken paths, the hint-aware protocol selects the longest living path within a dynamic neighbourhood. To achieve this goal, three metrics are computed to estimate the life duration of a path using heading and speed hints. Then, they are combined to provide a selection of paths that are supposed to be long-lived. After a simulation ran on vehicles data
in Boston, the paths selected, when taking into accounts those metrics, are two to five times longer than the median of all paths.

4.6 Limitations

As with every application running on mobile platforms, there are limitations. Energy is a big drawback and probing for hints can largely reduce the battery life of mobile devices. Some solutions exist, like using low energy sensors (i.e accelerometer), reducing sampling rate, triggering higher energy sensors only when needed and reverting to hint-unaware protocols when the battery is low. Calibration is another issue because of the diversity between sensors across different devices and platforms. Finally, sharing mobility hints might expose privacy concerns. A solution to this problem is to use encrypted hints.

5 Crowdsourcing

5.1 Definition and characteristics

Crowdsourcing comprehends a wide array of activities and is therefore hard to define precisely. There is a variety of definitions and no consensus on the different aspects forming crowdsourcing. In [13], Estellés-Arolas and al. determine from 209 documents, 8 characteristics defining a crowdsourcing project:

- There is a clearly defined crowd
- There exists a task with a clear goal
- The recompense received by the crowd is clear
- The crowdsourcer is clearly identified
- The compensation to be received by the crowdsourcer is clearly defined
- It is an online assigned process of participative type
- It uses an open call of variable extent
- It uses the Internet

Crowdsourcing can therefore be described as the process of obtaining the desired outcome, by using the abilities of a large community over the Internet, for a defined compensation. It is closely related to crowdsensing and the primary difference is the nature of the tasks performed. In crowdsourcing, a bigger task is often divided
in small tasks performed by humans. On the other hand, in crowdsensing, a lot of small sensing data is brought together to provide a service. Crowdsensing can be view as a crowdsourcing project where individuals share their sensing capabilities to achieve a given goal.

As described in [7], there are tasks that are still difficult to solve by software while they are easily executed by humans. These are for instance, tagging images, natural language processing, summarization and translation. These tasks are not location-based, and thus do not need a mobile platform. But one can imagine tasks that are dependant on the location like audio recording of a lecture, pictures and information of the surroundings of a location, real-time weather information. Therefore, there is a demand for a mobile platform able to provide human workforce to people who need the kind of services described.

5.2 Execution of the process

To better understand the different challenges associated with crowdsourcing applications, I describe in this section the execution of a general-purpose crowdsourcing process. A requester is defined as the person or organization who issue the task and a worker is defined as a person who provides his knowledge or resources to help solve the task. The process can be divided in four steps [26]:

5.2.1 Registration and task specification

First, the workers and the requesters register on the platform and provide their personal informations and qualifications. Then, the requester can specify the task to be crowsourced.

5.2.2 Task distribution

The task is promoted on the platform, displaying the rewards associated with the different requirements. Workers can discuss the approach with requesters and team up with other participants in order to accomplish the task.

5.2.3 Task resolution

Once requesters and workers have agreed on the different terms of the contract, the platform sets up the environment and the tools needed by the workers to resolve the task. The platform acts as a broker between the two parties and must allocate the resources and provide the infrastructure for the crowd to use.
5.2.4 Task evaluation

Once the workers have finished the request, the requester validates it against the requirements set before. After the request’s solution is accepted by the requester, the platform issues the payments and the participants can rate their collaboration.

5.3 Examples

In [7], researchers developed a platform that allows requesters to provide location-based tasks to workers. The system is composed of three parts: a web platform used by requesters to upload new tasks, a mobile client for workers to find and resolve tasks, and a server to keep and assign tasks to workers. The workers can retrieve tasks based either on their current location or on an address they specified. After two studies on eighteen participants, the results show that a location-based crowdsourcing platform is feasible. And their main conclusions were that: users prefer address-based task selection, picture tasks are more popular, tasks were mainly solved at or close to home and tasks were solved after work.

A paradigm of a well developed crowdsourcing platform is Amazon Mechanical Turk (MTurk). It offers to businesses an on-demand, scalable workforce and provides users with a selection of thousands of tasks [2]. A requester can create any task that can be accomplished on a computer on the platform and set a monetary reward (usually a few cents) for accomplishing a portion of the task. Workers can browse the tasks and choose the one they are interesting in by comparing the effort and the possible rewards. To ensure the requester they will obtain a high quality result, some tasks require qualifications, such as rating level.

MTurk is heavily used to transcribe images or audio samples to text. It has also been used for more unconventional tasks, for example to search for a missing boat by asking users to analyse satellite images [12]. However, MTurk is web based and not a purely mobile crowdsourcing application, which I choose to focus more on in this paper.

Other well-known examples of non-mobile crowdsourcing projects are Wikipedia and Linux, where volunteers contribute to develop a specific product. Given the loose definition of crowdsourcing, even website such as YouTube can be considered crowdsourcing examples [12]. In this example, a crowd builds a collection of videos that can be shared between users.
5.4 Challenges

Mobile crowdsourcing applications share many characteristics and challenges with crowdsensing applications. These challenges include, just to name a few, developing a common architecture, finding incentives and managing energy consumption. Other challenges more specific to crowdsourcing applications emerges such as finding what contributions can users make and how to combine them. A contribution can be easy, such as answering a question, while others can be very demanding, like resolving a tricky problem.

In [26], Vukovic, an IBM researcher, studied 22 existing crowdsourcing platforms and their applicability at an enterprise level. After defining the needed features and analysing how the current platforms fulfil them, he came up with a number of challenges that need to be addressed to build a general-purpose crowdsourcing application. The crowdsourcing request model must include complex criteria and the ability to ask providers different aspects of a request. An effective matchmaking process is needed, to attribute tasks to providers. Providers should have the ability to evaluate tasks based on complex criteria, not just incentives. The price should be dynamic and adapted according to the demand and providers availability. Virtual team formation should be based on the skill-set and social networks. Crowdsourcing services should be integrated in the cloud environment.

Some of these challenges, such as dynamic pricing, have been studied in [8]. By building effective incentive policies depending on the crowdsourcing project, the price can be dynamically adapted to ensure the best completion level for a given budget. However, as noted in Vukovic’s paper, current crowdsourcing platforms are often purpose-built and support only the range of tasks they were conceived for. To address these challenges, he has started to build his own crowdsourcing platform based on these observations.

6 Conclusion

As seen above, with all the possible applications, the crowdsensing and crowdsourcing fields are very wide and include a lot of other fields related to computer science such as data mining, data processing, storage, network, communication and much more. Although it has been used since several years already, mobile crowdcomputing is still very young and a lot of space for improvements is left. The new possibilities offered by the billions of devices forming a worldwide network are tremendous and I reckon crowdsensing holds great promises.
However, there are still many challenges to address, among which many of the technical ones will probably be irrelevant in the near future as devices become more powerful and last longer. But the human factor will still be a part of the equation, and motivation and privacy will remain big concerns. Yet, solutions to those challenges are coming, which, combined with an unifying architecture for all devices, will allow crowdsensing to develop much faster in the future.
References


[22] Suhas Mathur, Tong Jin, Nikhil Kasturirangan, Janani Chandrasekaran, Wenzhi Xue, Marco Gruteser, and Wade Trappe. “Parknet: drive-by sensing of


